CONTENTS

CH # 1 INTRODUCTION TO PROGRAMMING

1.1 PROGRAMMING ENVIRONMENT

LONG QUESTIONS

1. Explain Programming Environment in detail. (K.B+U.B)

Ans: A collection of all the necessary tools for programming makes up a programming environment. It is essential to setup a programming environment before we start writing programs. It works as a basic platform for us to write and execute programs.

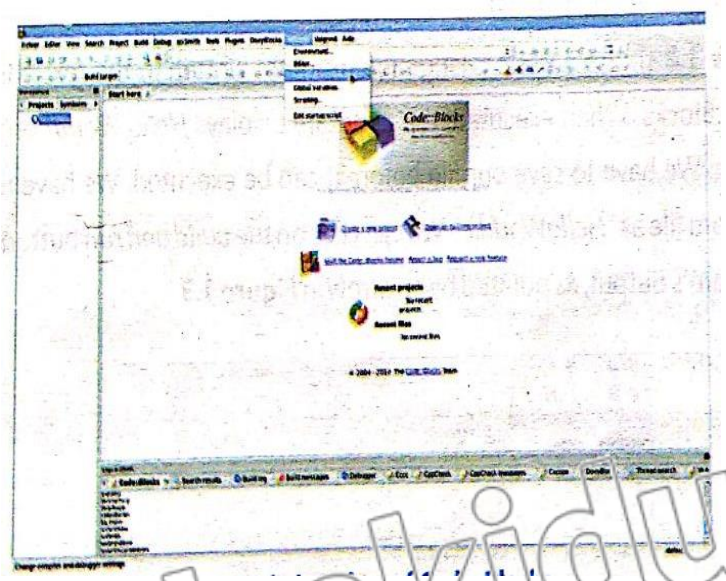
Integrated Development Environment (IDE):

A software that provides a programming environment to facilitate programmers in writing and executing computer programs is known as an Integrated Development Environment (**IDE**).

An **IDE** has a graphical user interface (**GUI**), meaning that a user can interact with it using windows and buttons to provide input and get output. An **IDE** consists of tools that help a programmer throughout the phases of writing, executing and testing a computer program. This is achieved by combining text editors, compilers and debuggers in a single interface.

Some types of the many available **IDEs** for C programming language are:

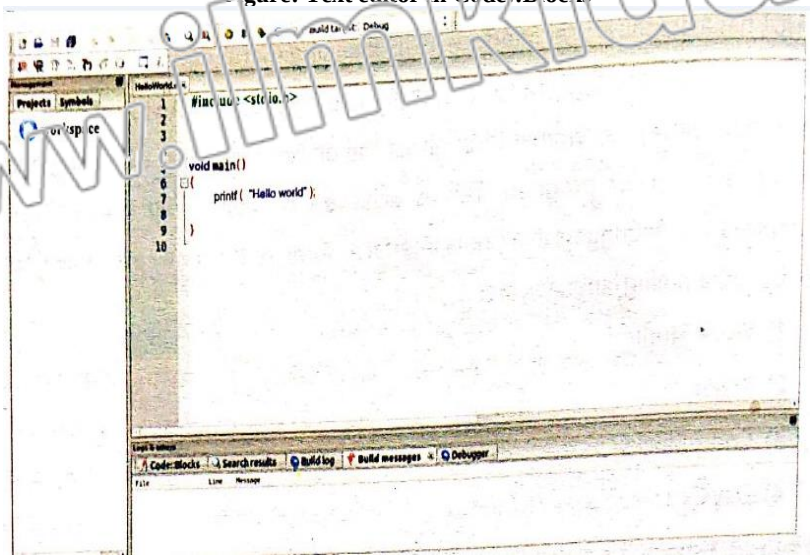
1. Visual Studio
2. Xcode
3. Code::Blocks
4. Dev C + +



Text Editor:

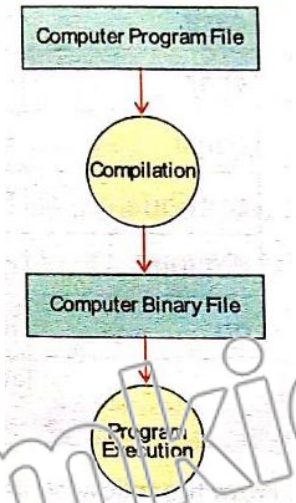
A text editor is a software that allows programmers to write and edit computer programs. All IDEs have their own specific text editors. It is the main screen of an IDE where we can write our programs.

Figure: Text editor in Code::Blocks



Compiler:

Computers only understand and work in machine language consisting of 0's and 1's. They require the conversion of a program written in programming language to machine language, in order to execute it. This is achieved using a compiler. A compiler is a software that is responsible for conversion of a computer program written in some high level programming language to machine language code.



SHORT QUESTIONS

Q.1 Define computer program or computer software. (K.B)

Ans: Computers need to be fed a series of instructions by humans which tell them how to perform a particular task. These series of instructions are known as a computer program or software.

Q.2 What is a programmer? (K.B)

Ans: The person who knows the detail about the syntax of the programming language and can write a program is called a programmer.

Q.3 What are programming languages? (K.B)

Ans: Computer programs are written in languages called programming languages. Some commonly known programming languages are

- Java
- C
- C+ +
- Python.

Q.4 Name any five programming language. (K.B)

Ans:

- C language
- C++
- C#
- JAVA
- Python

Q.5 Define programming Environment. (K.B)

Ans: A collection of all the necessary tools for programming makes up a programming environment. It is essential to setup a programming environment before we start writing programs. It works as a basic platform for us to write and execute programs.

Q.6 Define IDE. (K.B)

Ans: A software that provides a programming environment which facilitates the programmer in writing and executing computer programs is known as an Integrated Development Environment (IDE).

Q.7 Name some commonly available IDE of C. (K.B)

Ans:

- visual studio
- code:: blocks
- x code
- Dev C ++
- Turboo C

Q.8 Define Text editor. (K.B)

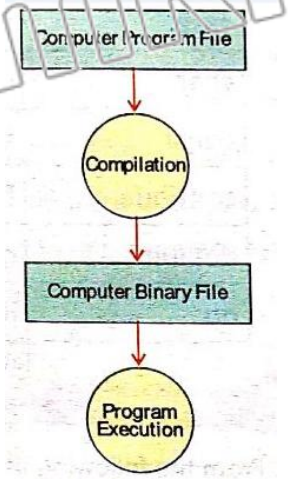
Ans: A text editor is a software that allows programmers to write and edit computer programs. All IDEs have their own specific editors.

Q.9 Define compiler. (K.B)

Ans: A compiler is a software that is responsible for conversion of a computer program written in some programming language to machine language code.

Q.10 Show program execution with help of flowchart diagram. (K.B+U.B)

Ans:



Q.11 How many modules do 'C' IDE has? (K.B+U.B)

Ans: C IDE has 5 modules.

- Text Editor
- Compiler
- Linker
- Loader
- Debugger

MULTIPLE CHOICE QUESTIONS

1. The process of writing a computer program in computer programming language to solve a particular problem is called _____. (K.B)

(A) Programming (B) Software Testing
(C) Software Maintenance (D) Software Documentation

2. Which of following is related to computer program? (K.B+U.B)

(A) Solve Particular Problem (B) Developed in Computer language
(C) Coding (D) All of Above

3. The computer languages that are used for writing program are called _____. (K.B)

(A) Natural Language (B) Programming Language
(C) Native Language (D) National Language

4. A set of instruction writes in programming language is called _____. (K.B)

(A) Program (B) Software Design (C) Algorithm (D) Flowchart

5. Each statement of programming language has its own _____. (K.B)

(A) Syntax (B) Semantic (C) Coding (D) Both 'A' & 'B'

6. Computer microprocessor execute to solve a particular problem _____. (K.B+U.B)

(A) Set of Instruction (B) Data (C) Algorithm (D) Flowchart

7. The rules of programming language according to which statement of a program is written, called as _____. (K.B)

(A) Syntax (B) Format (C) Semantic (D) Algorithm

8. **The grammatical rules of a natural language are similar to programming language. (K.B)**
(A) Algorithm (B) Pseudo Code (C) Documentation (D) Syntax

9. **A language that is understandable by computer is called: (K.B)**
(A) Native Language (B) National
(C) Programming Language (D) Local Language

10. **The use of computer programming language includes: (K.B+U.B)**
(A) Write Program
(B) Use for Communication between User and Computer
(C) Use to Control System
(D) All of Above

11. **The computer programming languages are classified into categories: (K.B)**
(A) Two (B) Three (C) Four (D) Five

12. **Some of the variable IDEs of 'C' are. (K.B)**
(A) Visual Studio (B) X code (C) code block (D) All

13. **A set of processes and programming tools used to develop computer program is called. (K.B+U.B)**
(A) Programming Environment (B) Programming Skills
(C) Programming Tools (D) Programming Techniques

14. **To create, compile and run a program in programming language we use. (K.B)**
(A) IDE (B) Standard Tools (C) GUI (D) Text Editor

15. **In C language IDE consist of the modules: (K.B)**
(A) Two (B) Four (C) Three (D) Five

16. **A simple word processor that is used to create and edit source code of a program. (K.B)**
(A) Text Editor (B) Word Processor (C) Microsoft Word (D) Spread Sheet

17. **A computer software that translates 'C' language program into machine code. (K.B)**
(A) Compiler (B) Interpreter (C) Linker (D) Assembler

18. **Which one of the following is not an IDE of 'C'? (K.B)**
(A) Visual studio (B) X code (C) Dev C # (D) C #

19. **GUI stands for (K.B+U.B)**
(A) Graphical Use Internet (B) Good Using Interface
(C) Graphical User Interface (D) Graphical Using Internet

20. **IDE allows a user to interact with it using windows and button for inputs and outputs. (K.B)**
(A) GUI (B) CLI (C) MDI (D) All of these

21. **provides an interface to a user. (K.B)**
(A) IDE (B) Text Editor (C) Compiler (D) None of these

22. **IDE is a combination of. (K.B+U.B)**
(A) Text Editors (B) Compiler (C) Debuggers (D) All of these

23. **Some commonly used programming languages are? (K.B)**
(A) C (B) C++ (C) C# (D) All of these

24. **C is a _____ language. (K.B)**
(A) Basic (B) High level (C) Programming (D) Both B and C

25. **allows a programmer to write and edit a program. (K.B)**
(A) Text editor (B) compiler (C) IDE (D) All of these

26. **All IDE's have ____ text Editors. (K.B)**
(A) Same (B) Unique (C) Both A & B (D) None of these

27. **is the main screen of IDE. (K.B)**
(A) Text Editor (B) Compiler (C) Debugged (D) None of these

28. Computers only understands _____. **(K.B)**
(A) 0 and 1 (B) English (C) Roman language (D) None of these

29. _____ converts a programming language to machine code. **(K.B)**
(A) Compiler (B) Text Editor (C) Debugger (D) All of these

30. C language uses a _____. **(K.B+U.B)**
(A) Compiler (B) Interpreter (C) Both A & B (D) None of these

31. Program written in high level language is called _____. **(K.B)**
(A) Source code (B) Object code (C) Both A & B (D) None of these

32. Program written in low level language is called _____. **(K.B)**
(A) Source code (B) Object code (C) Both A & B (D) None of these

1.2 PROGRAMMING BASIC

SHORT QUESTIONS

Q.1 Define syntax. **(K.B)**

Ans: Every programming language has some primitive building blocks and follows some grammar rules known as its syntax.

Q.2 Define syntax error with the help of example. **(K.B)**

Ans: While programming, if proper syntax or rules of the programming language are not followed, the program does not get compiled. In this case, the compiler generates an error. This kind of errors are called syntax errors.

Example:
Typing **'pintf'** instead of **'printf'**

MULTIPLE CHOICE QUESTIONS

1. ____ is the set of rules to write a program. **(K.B)**
(A) syntax (B) semantic (C) X code (D) None of these

2. Program does not gets compiled in case of _____ errors. **(K.B)**
(A) Syntax (B) Logical (C) Both (D) None of these

3. 5=a is a type of _____ error. **(K.B+U.B)**
(A) Syntax (B) Logical (C) Routine (D) None of these

4. Syntax errors are also known as ____ errors. **(K.B)**
(A) Typographical (B) Exaction (C) Silly (D) Del

1.2.1 RESERVED WORDS

SHORT QUESTIONS

Q.1 Define reserved words. **(K.B)**

Ans: Every programming language has a list of words that are predefined. Each word has its specific meaning already known to the compiler. These words are known as reserved words or keywords.

Q.2 How many keywords a C language have? **(K.B)**

Ans: C language has 32 reserved words

Q.3 Name any five reserved words **(K.B)**

Ans:

- if
- int
- auto
- break
- do

Q.4 Write all 32 reserved words of C. **(K.B)**

Ans:

auto	Double	int	struct
break	Else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

MULTIPLE CHOICE QUESITONS

1. In 'C' language word that are part of programming language and have special use: **(K.B+U.B)**

(A) Special Character (B) Digits

(C) Constants (D) Reserve Words

2. The number of reserved words used in 'C' language are _____. **(K.B)**

(A) 20 (B) 32 (C) 31 (D) 42

3. Which one of the following is not a reserve word of 'C' language? **(K.B)**

(A) auto (B) char (C) int (D) x

4. Reserved words are also called _____. **(K.B)**

(A) Keyboards (B) Predefined (C) Both 'A' & 'B' (D) None of these

1.2.2 STRUCTURE OF C PROGRAM

LONG QUESTIONS

1. Explain structure of 'C' program. **(K.B+U.B)**

Ans:

STRUCTURE OF A 'C' PROGRAM

A 'C' program is divided into three main parts:

1. **Link section or header section:** While writing programs in 'C' language, we make extensive use of functions that are already defined in the language. But before using the existing functions, we need to include the files where these functions have been defined. These files are called header files. We include these header files in our program by writing the include statements at the top of program.
   General structure of an include statement is as follows:
   #include<header_file_name>
2. **Main section:** It consists of a **main ()** function. Every 'C' program must contain a main () function and it is the starting point of execution.
3. **Body of main () function:**
   The body of main () is enclosed in the curly braces {}. All the statements inside the curly braces make the body of main function.
   Example:

```
# include <stdio.h>
void main ( )
```

```
{
printf ("Hello World!");
}
```

In the above program the statement **printf ("Hello World!");** uses a predefined function printf to display the statement **Hello World!** on computer screen. We can also create other functions in our program and use them inside the body of **main ()** function.

SHORT QUESTIONS

Q.1 How many parts do a C program have? (K.B)

Ans: C program has three parts

1. Link section or header section
2. Main section
3. Body of main ( ) function

Q.2 Define Link section or header section. (K.B)

Ans: While writing programs in 'C' language, we make extensive use of functions that are already defined in the language. But before using the existing functions, we need to include the files where these functions have been defined. These files are called header files. We include these header files in our program by writing the include statements at the top of program.

General structure of an include statement is as follows:

#include<header_file_name>

Example:

include <stdio.h>

Q.3 Define Main section of C program. (K.B)

Ans: It consists of a main () function. Every 'C' program must contain a main () function and it is the starting point of execution.

Example:

void main ( )

Q.4 What is body of Main () function? (K.B)

Ans: The body of **main ()** is enclosed in the curly braces **{}**. All the statements inside the curly braces make the body of main function.

Example:

```
{
printf ("Hello World!");
}
```

In the above program, the statement *printf* ("Hello World!"); uses a predefined function *printf* to display the statement Hello World! on computer screen.

Q.5 What points should be kept in mind while developing a C program? (K.B+U.B)

Ans: Following points must be kept in mind in order to write syntactically correct C language programs.

- The sequence of statements in a C language program should be according to the sequence in which we want our program to be executed.
- C language is case sensitive. It means that if a keyword is defined with all small case letters, we cannot capitalize any letter i.e. int is different from int. Former is a keyword, whereas latter is not.
- Each statement ends with a **semi-colon(;)** symbol.

Q.6 Identify different parts of the following C program. (U.B)

Ans:

Program	Parts of C program

#include <stdio.h>
#include <conio.h>
void main()
{
printf("I am a student of class 10");
getch ( );
}
	//the processor directives
	//preprocessor directives
	//main function
	//body of fiction {} are known as delimiters
	"); //printfunction // ("") referred as string literals
	// input function

MULTIPLE CHOICE QUESITON

1. How many types of header files are there in 'C' language program? (K.B)
(A) 2 (B) 4 (C) 3 (D) Many

2. "math.h" header file contains function: (K.B)
(A) printf ( ) (b) getch ( ) (c) clrscr ( ) (d) sin ( )

3. Which one of the following is included in "conio.h" header file of 'C'? (K.B)
(A) getche ( ) (b) scanf ( ) (c) pow ( ) (d) log ( )

4. Which one of the following is not a part of 'C' program? (K.B+U.B)
(A) Preprocessor Directives (B) Body of Program
(C) Main ( ) Function (D) Compiler

5. In 'C' program the instructions for compiler is called: (K.B)
(A) Preprocessor Directive (B) Body of main ( )
(C) Main ( ) Function (D) Syntax

6. In 'C' language preprocessor directive starts with a ____. (K.B)
(A) ; (B) # (C) : (D) !

7. The most commonly used preprocessor directive includes: (K.B)
(A) Includes (B) Header File (C) Define (D) Both 'A' & 'C'

8. In 'C' program the body of main function surrounded by: (K.B)
(A) ( ) (B) { } (C) [ ] (D) !

9. There are _____ sectors of a 'C' program. (K.B)
(A) 1 (B) 2 (C) 3 (D) 4

10. "stdio.h" consists of basic _____ functions. (K.B)
(A) Input (B) Output (C) Both 'A' & 'B' (D) None of these

11. _____ is enclosed in curly braces. (K.B)
(A) Body of main (B) Main (C) Link section (D) None of these

12. Syntax of header file. (K.B)
(A) #include <header – file –name> (B) # include <header >
(C) <header-file-name> (D) None of these

13. Starting point of execution is _____. (K.B+U.B)
(A) Main () function (B) Header file (C) link (D) None of these

14. Every 'C' program must contain _____. (K.B+U.B)
(A) Main (B) Link (C) Body of Main () (D) Both B & C

15. 'C' program follows ___ order for execution (K.B)
(A) Sequential (B) Decently (C) Selection (D) None of these

16. 'C' is a ____ language. (K.B+U.B)
(A) Case sensitive (B) Not case sensitive (C) Modular (D) None of these

17. In 'C' language a ____ A (K.B+U.B)
(A) = (B) ≠ (C) sometime (D) None of these

18. Each statement ends with _____. (K.B+U.B)
(A) : (B) ; (C) , (D) #

19. Semicolon(;) is used at the end of ____ statement. (K.B+U.B+A.B)
(A) Every (B) First (C) Last (D) None of these

20. Anything after _____ on the same line is considered a single line comment.(K.B+U.B)
(A) / (B) // (C) * (D) None of these

21. Anything after // on the same line is considered a _____. (K.B+U.B)
(A) Rule (B) Comment (C) Single line comment (D) Multiline comment

22. Multiline comments starts with ____. (K.B)
(A) */ (B) /* (C) // (D) *

23. Anything between /* and */ is considered _____. (K.B)
(A) Very words (B) Comment (C) Reserve words (D) None of these

1.2.3 COMMENTS IN 'C'

LONG QUESTIONS

1. Define Comments in 'C' language program? Explain its types (K.B)

Ans:

PURPOSE AND SYNTAX OF COMMENTS IN 'C' PROGRAMS

Comments are the statements in a program that are ignored by the compiler and do not get executed. Usually comments are written in natural language e.g. in English language, in order to provide description of our code.

Purpose of writing comments:

Comments can be thought of as documentation of the program. Their purpose is twofold.

1) They facilitate other programmers to understand our code.

2) They help us to understand our own code even after years of writing it.

We do not want these statements to be executed, because it may cause syntax error as the statements are written in natural language.

Syntax of writing comments:

aIn C programming language, there are two types of comments.

1. Single-line Comments

2. Multi-line Comments

Single-line comments start with //. Anything after **//** on the same line, is considered a comment. For example, **//** This is a comment.

Multi-line comments start with /* and end at */. Anything between **/*** and ***/** is considered a comment, even on multiple lines. For example,

/* this is
a multi-line
comment ***/**

Following example code demonstrates the usage of comments.

Example:

```
#include <stdio.h>
/*this program displays "I am a student of class 10" on the output screen*/
void main( )
{ //body of main function starts from here printf ("I am a student of class 10");
} //body of main function ends here
```

SHORT QUESTIONS

Q.1 Define comments in a 'C' language program. (K.B)

Ans: Comments are the statements in a program that are ignored by the compiler and do not get executed. Usually comments are written in natural language e.g. in English language, in order to provide description of the code. Comments are used to increase the readability of the program.

TYPES OF COMMENT:

- Single-line comments
- Multi-line comments

Q.2 What are single line comments? (K.B)

Ans: Start with //. Anything after // On the same line, is considered to be a comment. For example, //This is a comment.

Example:

```
// This is a comment.
```

Q.3 What are multiple comments? (K.B)

Ans: start with /* and end at */. Anything between /* and */ is considered a comment, even on multiple lines. For example,

Example:

```
/*This is
a multi-line Comment*/
```

Q.4 How many types of comments can be used in 'C'? (K.B+U.B)

Ans: 1- Single-line comments
2- Multi-line comments

Q.5 Differentiate between single line and multiple comments. (K.B+U.B)

Ans: The differences between single line and multiline comment are as follows:

Single Line Comment	Multiline Comment
• The // is used as single line comment.	• The /*….*/ is used for multiple line comments.
• The syntax of single line comment is: // comment	• The syntax of multiline comment is: /*……..*/
Example: • Comments comprise on one line. • // Hello to the world	**Example:** • Comment can span to multiple lines: • /* Welcome To the C language This is my first program*/

Q.6 Write a program that uses comments. (A.B)

Ans:

EXAMPLE CODE

```
/*This program displays "I am a student of class 10"
on the output screen*/
#include <stdio.h>
void main( )
{ //body of main function starts from here
printf ("I am a student of class 10");
} //body of main function ends here
```

Q.7 **(A.B)**

ACTIVITY 1.5

Tick valid comments among the following:

- *comment goes here*
- **/comment goes here/ → invalid**
- %comment goes here%
- /* comment goes here*/ **→ valid**
- /* comment goes here/
- //comment goes here*/

MULTIPLE CHOICE QUESTIONS

1. Which is related to 'C' comments? (K.B+U.B)
(A) Single Line (B) Multiple Line (C) Statement Terminator (D) Both 'A' & 'B'

2. It is good programming practice to add in C program: (K.B)
(A) Comment/s (B) Question Marks (C) Variable (D) Constant

3. Which one of the following is used to show single line comments: (K.B+U.B)
(A) // (B) || (C)\\ (D) !

4. Multi line comment in C language represented as: (K.B+U.B)
(A) // (B) */*/ (C) /**/ (D) *//*

5. _____ are statements in a program that are ignored by the compiler. (K.B)
(A) Keywords (B) Reserve words (C) Comments (D) None of these

6. Compiler ignores _____. (K.B)
(A) Reserve words (B) Comments (C) Keywords (D) None of these

7. Comments are the statement that are _____. (K.B)
(A) not executed (B) ignored (C) Both 'A' & 'B' (D) Produced

8. Usually comments are written in _____. (K.B)
(A) English Language (B) Natural Language (C) Both 'A' & 'B' (D) None of these

9. Usually _____ are written in natural language. (K.B)
(A) Keywords (B) Reserve words (C) Comments (D) None of these

10. Comments provide _____. (K.B)
(A) Precaution of code (B) Description of code
(C) Syntax of code (D) All of these

11. There are _____ types of writing comments. (K.B)
(A) 2 (B) 3 (C) 4 (D) 5

12. _____ facilitate other programmers to understand own code. (K.B)
(A) Keywords (B) Reserve words (C) Comments (D) None of these

13. Comments are not executed as _____ may occur. (K.B)
(A) Routine error (B) Syntax error (C) Comments (D) None of these

14. Single-line comments starts with ___. (K.B+U.B)
(A) // (B) */ (C) (D) *

15. _____ comments starts with // (K.B+U.B)
(A)Multiline comment (B) Single-line (C) Both 'A' & 'B' (D) None of these

1.3 CONSTANTS AND VARIABLES
1.3.1 CONSTANTS
1.3.2 VARIABLES

LONG QUESTIONS

1. Define Constants? Explain its types. (K.B+U.B)

Ans: Constants are the values that cannot be changed by a program e.g. 5, 75.7, 1500 etc. In C language primarily we have three types of constants:

1. **Integer Constants:** These are the values without a decimal point e.g. 7, 1256, 30100, 55555, -54, -2349 etc. They can be positive or negative. If the value is not preceded by a sign, it is considered as positive.
2. **Real Constants:** These are the values including a decimal point e.g. 3.14, 15.3333, 75.0, -1575.76, -7941.2345 etc. They can also be positive or negative.
3. **Character Constants:** Any Single small case letter, upper case letter, digit, punctuation mark, special symbol enclosed within ' ' is considered a character constant e.g. '5', '7', 'a', 'X', '! ', '; ' etc. A digit used as a character constant i.e. 9, is different from a digit used as an integer constant i.e. 9. We can add two integer constants to get the obvious mathematical result e.g. 9 + 8 = 17, but we cannot add a character constant to another character constant to get the obvious mathematical result e.g. '9' + '8' $\neq$ 17.

2. Define variables? Explain its types. (K.B+U.B)

Ans: A variable is a symbolic name that represents a value that can change during execution of a program. A variable has a name, known as variable name and it holds data of other types. A number or any other types of data held in a variable is called its value. It also indicates the types of value variable can represent.

Variables are of two types:

- Numeric Variables
- Character Variables

Numeric Variables:

Numeric variables are used to represent numeric values in computer programs. They represent an integer and floating-point values.

Examples:

- Sum
- Avg
- Length
- Salary
- Marks

Character Variables:

Character variables represent character values in computer programs. It can represent a single character or a string of characters.

Examples:

- Name
- City
- Gender

Explanation:

When a variable is used in a computer program, the computer associates it with a particular memory location. The value of a variable at any time is the value stored in the associated memory location at that time. Variables are used so that the same space in memory can hold different values at different times.

SHORT QUESTIONS

Q.1 Define character set of 'C' language. (K.B)

Ans: Each language has a basic set of alphabets (character set) that are combined in an allowable manner to form words, and then these words can be used to form sentences. Similarly in C programming language we have a character set that includes:

1) Alphabets (A, B, …., Y, Z), (a, b …. y, z)
2) Digits (0 - 9)
3) Special symbols (~ '!@#% ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /)

Q.2 Define constants of 'C' language. (K.B)

Ans: Constants are the values that cannot be changed by a program e.g. 5, 75.7, 1500 etc. In C language, primarily we have three types of constants:

Types of constant:

- Integer constants
- Real constants
- Character constants

Q.3 Define Integer constant. (K.B)

Ans: These are the values without a decimal point e.g. 7, 1256, 30100, 55555, -54, -2349 etc. They can be positive or negative. If the value is not preceded by a sign, it is considered as positive.

Q.4 Define real constants. (K.B)

Ans: These are the values including a decimal point e.g. 3.14, 15.3333, 75.0, -1575.76, -7941.2345 etc. They can also be positive or negative.

Q.5 Define character constant. (K.B)

Ans: Any single small case letter, upper case letter, digit, punctuation mark, special symbol enclosed within ' ' is considered a character constant e.g. '5', '7', 'a', 'X', '! ', '; ' etc.

Q.6 Differentiate between real constant and integer constant. (K.B + U.B)

Ans:

Real Constant	Integer Constant
• These are the values including a decimal point	• These are the values without a decimal point
• They can also be positive or negative.	• They can be positive or negative. • If the value is not preceded by a sign, it is considered as positive.
Example: • 3.14, 15.3333, 75.0, -1575.76, -7941.2345	**Example:** • 7, 1256, 30100, 55555, -54, -2349

ACTIVITY 1.6 (A,B)

Identify the type of constant for each of the following values.

Solution:

12 → integer constant
1.2 → real constant
'*' → character constant
-21 → integer constant
32.768 → real constant

'a' → character constant
-12.3 → real constant
41 → integer constant
40.0 → real constant
'1' → character constant

Q.7 Tell a type of constant from the following list. (K.B+U.B+A.B)

12
1.2
'*'
-21
32.768
'a'
-12.3
41
40.0
'1'

Ans

Constant	Type of Constant
12	Integer constant
1.2	Real constant
'*'	Character constant
-21	Integer constant
32.768	Real constant
'a'	Character constant
-12.3	Real constant
41	Integer constant
40.0	Real constant
'1'	Character constant

Q.8 Define variables. (K.B)

Ans: A variable is actually a name given to a memory location, as the data is physically stored inside the computer's memory. The value of a variable can be changed in a program. It means that, in a program, if a variable contains value 5, then later we can give it another value that replaces the value 5. Each variable has a unique name called identifier and has a data type.

Q.9 Write down any three names of data type. (K.B+U.B+A.B)

Ans:

Type of Data	Matching Data Type in C language	Sample Values
Integer	int	123
Real	float	23.5
Character	char	'a'

Each variable in C language has a data type. The data type not only describes the type of data to be stored inside the variable but also the number of bytes that the compiler needs to reserve for data storage.

MULTIPLE CHOICE QUESIONS

1. Each language has a basic set of alphabets_____. (K.B)
(A) Comments (B) Keyboards (C) Character Set (D) Variable

2. In c programming we have a charter set that includes ____. (K.B+U.B)
(A) Alphabets (B) Digits (C) Special Symbols (D) All of these

3. How many digits in character set are available ranging from _____. (K.B)
(A) 1-9 (B) 0-9 (C) 1-10 (D) 0-10

4. Constant and variable are used in program to write: (K.B)
(A) Expression (B) Equation (C) Statement (D) Semantic

5. The quantities whose value do not change during program execution is called. (K.B)
(A) Variable (B) Constant (C) Reserve word (D) Expression

6. Which one of the following is not a type of constant? (K.B)
(A) Numeric (B) String (C) Reserve Word (D) Both A & B

7. How many types of numeric constant? (K.B)
(A) 2 (B) 4 (C) 3 (D) 5

8. Which one of the following is an example of floating-point numeric constants? (K.B+U.B)
(A) 7145 (B) -234 (C) 166.75 (D) 26

9. String constant are written with in: (K.B)
(A) "" (B) ( ) (C) [ ] (D) { }

10. Which one of the following is an example of character constant is: (K.B+U.B)
(A) 'a' (B) (A) (C) "a" (D) {a}

11. A symbolic name that represents a value that can change during execution of program is called: (K.B+U.B)
(A) Constant (B) Variable (C) String (D) Reserve Word

12. Which one is an example of real constant. (K.B+U.B)
(A) 15.33 (B) -37.8 (C) 37 (D) Both A and B

13. Character constant are always written inside. (K.B)
(A) "" (B) ** (C) , (D) 1,'

14. Sum, avg, length, salary and marks are examples of: (K.B+U.B)
(A) Numeric Constant (B) Data Set
(C) Numeric Variable (D) Character Constant

15. Example of character variable includes: (K.B+U.B)
(A) Name (B) Gender (C) City (D) All of these

16. A variable used in a computer program associates with a: (K.B+U.B+A.B)
(A) Device (B) Constant (C) Memory location (D) Expression

17. In C program, the length of variable name has _____ significant characters. (K.B+U.B)
(A) 21 (B) 31 (C) 30 (D) 32

18. Which one of the following cannot be used as a variable? (K.B+U.B)
(A) Special character (B) Character (C) Digits (D) Variable

19. In C language which of following cannot be used as variable? (K.B+U.B)
(A) auto (B) if (C) int (D) All of these

20. Which one is a correct example of variable name? (K.B+U.B)
(A) Mass (B) Over Time (C) StuID (D) Book ID

21. Each variable has a _____. (K.B+U.B)
(A) Unique Name (B) Identifier (C) Data Type (D) All of these

1.3.3 DATA TYPES OF A VARIABLES

SHORT QUESTIONS

Q.1 Define data types. (K.B)

Ans: Each variable in C language has a data type. The data type not only describes the type of data to be stored inside the variable but also the number of bytes that the compiler needs to reserve for data storage.

- Integer
- Float
- Character

Q.2 Define integer data type. (K.B)

Ans: Integer data type is used to store integer values (whole numbers). Integer takes up 4 bytes of memory. To declare a variable of type integer, we use the keyword int.

Signed int: A signed int can store both positive and negative values ranging from -2, 147, 483, 648 to 2,147, 483, 647. By default. Type int is considered as a signed integer.

Unsigned int: An unsigned int can store only positive values and its value ranges from 0 to +4,292,967,295. Keyword unsigned int is used to declare an unsigned integer.

Q.3 Define signed integer. (K.B)

Ans: A signed int can store both positive and negative values ranging from -2, 147, 483, 648 to 2,147, 483, 647. By default type 'int' is considered as a signed integer.

Q.4 Define unsigned integer. (K.B)

Ans: An unsigned into can store only positive values and its value ranges from 0 to +4,292,967,295. Keyword unsigned int is used to declare an unsigned integer.

Q.5 Define float data type. (K.B)

Ans: Float data type is used to store a real number (number with floating point) up to six digits of precision. To declare a variable of type float, we use the keyword float. A float uses 4 bytes of memory. Its value ranges from 3.4×10^{-38} to 3.4×10^{38}.

Q.6 Define character data type. (K.B)

Ans: To declare character type variables in C, we use the keyword char. It takes up just 1 byte of memory for storage. A variable of type char can store one character only.

Q.7 Differentiate between integer and floats data types. (K.B+U.B)

Ans:

Integer	Float
• Integer data type is used to store integer values (whole numbers).	• Float data type is used to store a real number (number with floating point) up to six digits of precision.
• Integer takes up 4 bytes of memory. • To declare a variable of type integer, we use the keyword **int**.	• A float uses 4 bytes of memory. • To declare a variable of type float, we use the keyword **float**.
Example: • -32,768 to +32,767	**Example:** • 3.4×10^{-38} to 3.4×10^{38}

MULTIPE CHOICE QUESTIONS

1. How many data types provided by 'C' language? (K.B)
(A) 2 (B) 4 (C) 3 (D) 5

2. In 'C' language int and short int data type occupy memory? (K.B)
(A) 1 Byte (B) 2 Bytes (C) 3 Bytes (D) 4 Bytes

3. In 'C' which data types is not used: (K.B)
(A) short int (B) float (C) unsigned int (D) float int

4. Float data type in 'C' occupy memory space: (K.B)
(A) 2 Bytes (B) 4 Bytes (C) 8 Bytes (D) 70 Bytes

5. In 'C' language a character variable can only store one. (K.B)
(A) Digit (B) Character (C) Byte (D) Word

6. In 'C' language unsigned integer data type have range. (K.B)
(A) 0-65535 (B) $10.^{0888}$-10^{308} (C) 0-4,294,697,295 (D) 0-32768

7. In 'C' language single integer data type have range. (K.B)
(A) -22,147,483,648 102,147,483,647 (B) -327651032765
(C) 0.56655 (D) -2,147,483,648 – 2,147,483647

8. Character data type occupy memory space. (K.B)
(A) 3 bytes (B) 1 byte (C) 4 bytes (D) 0 byte

1.3.4 NAME OF A VARIABLE

LONG QUESTIONS

1. Explain rules for naming variables in 'C' language. (K.B+U.B)

Ans: Rules For Naming Variables in 'C' Language

The following are the rules for specifying variable names in C language.

- A variable begins with a letter or underscore and may consist of **letters, underscores "_" and/or digits.**
- The underscore may be used to improve readability of the variable name. For example, over_time.
- There is no restriction on the length of a variable name. However, only the first 31 characters of a variable are significant. This means that if two variables have the same first 31 characters, they are considered to be the same variables.
- Both upper and lower-case letters are allowed in naming variables. An upper-case letter from Avg or avg.
- Special characters cannot be used as variable name e.g, #, ?, @, etc.
- Reserved words of C language such as int, case, if, etc, cannot be used as variable name.
- There must be no embedded blank in the name of variable. E.g father's name.

SHORT QUESTIONS

Q.1 Write down any three rules for naming a variable in 'C' language (K.B)

Ans: Each variable must have a unique name or identifier. Following rules are used to name a variable.

1. A variable name can only contain alphabets (uppercase or lowercase), digits and underscore (_) sign.
2. Variable name must begin with a letter or an underscore, it cannot begin with a digit.
3. A reserved word cannot be used as a variable name.

Q.2 ACTIVITY 1.7 (A.B)

ACTIVITY 1.7

Encircle the valid variable names among the following:

Hello,	1var	roll_num → valid	Air23Blue→valid	float
Case	$car	name → valid	=color	Float

MULTIPLE CHOICE QUESTION

1. A variable name can only contain. (K.B)

(A) Digits (B) Alphabets (C) Underscore (D) All of these

2. We should give appropriate name to a variable that desirable its ____. (K.B)

(A) Evolution (B) Purpose (C) Order (D) Type

1.3.5 VARIABLE DECLARATION

SHORT QUESTIONS

Q.1 Define variable declaration. (K.B)

OR

What is variable declaration?

Ans: We need to declare a variable before we can use it in the program. Declaring a variable includes specifying its data type and giving it a valid name. Following syntax can be followed to declare a variable.

Some examples of valid variable declarations are as follows:

```
unsigned int age;
float height;
int salary;
char marital_status;
```

MULTIPLE CHOICE QUESITONS

1. We need to declare a variable _____ it use. (K.B+U.B)

(A) After (B) Before (C) Compile (D) None of these

2. Declaring a variable includes specifying its_____. (K.B)

(A) Data Type (B) Valid Name (C) Both A & B (D) None of these

3. Multiple variable of _____ data type many also be declared in a single standard. (K.B)

(A) Different (B) Same (C) Both A & B (D) None of These

4. A variable cannot be declared unless we mention its _____. (K.B)

(A) Format (B) Data Type (C) Range (D) None of these

5. After declaring a variable its data type _____ be changed. (K.B+U.B)

(A) Can (B) Cannot (C) Both A & B (D) None of these

1.3.6 VARIABLE INITIALIZATION

SHORT QUESTIONS

Q.1 Define variable initialization. (K.B)

OR

What is variable initialization?

Ans: Assigning value to a variable for the first time is called variable initialization. **C** language allows us to initialize a variable both at the time of declaration, and after declaring it. For initializing a variable at the time of declaration, we use the following general structure.

data_type variable_name = value;

Q.2 What is the difference between variable declaration and variable initialization?(K.B + U.B)

Ans:

Variable Declaration:	Variable Initialization

We need to declare a variable before we can use it in the program. Declaring a variable includes specifying its data type and giving it a valid name.
following syntax can be followed to declare a variable.
data_type variable_name;
Example:
Some examples of valid variable declarations are as follows:
unsigned int age;
float height;
int salary;
char marital_status;
Multiple variables of same data type may also be declared in a single statement, as shown in the following examples:
unsigned int age, basic_salary,gross_salary;
int points_scored, steps;
	Assigning value to a variable for the first time is called variable initialization. C language allows us to initialize a variable both at the time of declaration, and after declaring it.
	For initializing a variable at the time of declaration, we use the following general structure.
	data_type variable_name = value;
	Example:
	Following example shows a program that demonstrates the declaration and initialization of two variables.
	#include<stdio.h>
	void main ()
	{
	char grade; //Variable grade is declared
	int value = 25; /*Variable value is declared and initialized.*/
	grade = 'A'; // Variable grade is initialized
	}

Q.3 Write a program that declares variables of appropriate data types to store your personal data. Initialize these variables with the following data: (A.B)

- **Initial letter of your name**
- **Initial letter of your gender**
- **Your age**
- **Your marks in 8th class**
- **Your height**

Ans:

ACTIVITY 1.8

Solution

```
#include < stdio.h>
#include < conio.h>
void main ()
{
char name = 'A';       // initial letter of your name
char gender = 'M';     // initial letter of your gender
int age = 26;          // your age
int marks = 500;       // marks in 8th class
float height =5.8;     // your height
}
```

MULTIPLE CHOICE QUESTIONS

1. Assigning value to a variable for the first time is called variable. (K.B)

(A) Declaration (B) Initialization (C) Compilation (D) None of these

2. For initializing a variable at the time of declaration, we use: (K.B)

(A) Data Type (B) Variable Name (C) Both A & B (D) None of these

EXERCISE

Q.1 Multiple Choice Questions

1) A software that facilitates programmers in writing computer program known as______.(K.B)
(A) a complier (B) an editor (C) an IDE (D) a debugger

2) ______is a software that is responsible for the conversion of program files to machine understandable and executable code. (K.B+U.B)
(A) Compiler (B) Editor (C) IDE (D) Debugger

3) Every programming language has some primitive building blocks and follows some grammar rules known as its _____. (K.B+U.B)
(A) Programming Rules (B) Syntax (C) Building Blocks (D) Semantic Rules

4) A list of words that are predefine and must not be used by the programmer to name his own variables are known as _____. (K.B+U.B)
(A) Auto Words (B) Reserved Words (C) Restricted Words (D) Predefined Words

5) Include statements are written in _____ section. (K.B)
(A) Header (B) Main (C) Comments (D) Print

6) _____ are added in the source code to further explain the techniques and algorithms used by the programmer. (K.B)
(A) Messages (B) Hints (C) Comments (D) Explanations

7) _____ are the values that do not change during the whole execution of program. (K.B)
(A) Variables (B) Constants (C) Strings (D) Comments

8) A float uses___ bytes of memory. (K.B)
(A) 3 (B) 4 (C) 5 (D) 6

9) For initializing variable, we use ____ operator. (K.B)
(A) → (B) = (C) @ (D)?

10) _____ can be thought of as a container to store constants. (K.B)
(A) Box (B) Jar (C) Variable (D) Collection

ANSWER KEY

1	C	6	C
2	A	7	B
3	B	8	B
4	B	9	B
5	A	10	C

Q.2 True of False

1) An IDE combines text editors, libraries, compilers and debuggers in a single interface. **(K.B)** T/F
2) Computers require the conversion of the code written in program file to machine language in order to execute it. **(K.B+U.B)** T/F
3) Column in reserved word in C programming language. **(K.B)** T/F
4) *comment goes here* is a valid comment. **(K.B)** T/F
5) Float can store a real number upto six digits or precision. **(K.B)** T/F

Q.3 Define the following. (K.B)

1) IDE
Ans: A software that provides a programming environment, which facilitates the programmer in writing and executing computer programs, is known as an Integrated Development Environment (IDE).

2) Compiler
Ans: A compiler is a software that is responsible for conversion of a computer program written in some programming language to machine language code and vice versa.

3) Reserved Words

Ans: Every programming language has a list of words that are predefined. Each word has its specific meaning already known to the compiler. These words are known as reserved words or keywords. If a programmer gives them a definition of his own, it causes a syntax error. Table shows the list of reserved words in C programming language. There are 32 reserved words in C.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

4) Main section of a program

Ans: It consists of a main () function. Every C program must contain a main () function and it is the starting point of execution.

5) char data type

Ans: To declare character type variables in C, we use the keyword char. It takes up just 1 byte of memory for storage. A variable of type char can store one character only.

Q.4 Briefly answer the following questions. (K.B+U.B)

1) Why do we need a programming environment?

Ans: In order to correctly perform any task, we need to have proper tools. For example, for gardening we need gardening tools and for painting we need a collection of paints, brushes and canvas. Similarly, we need proper tools for programming. A collection of all the necessary tools for programming makes up a programming environment. It is essential to setup a programming environment before we start writing programs. It works as a basic platform for us to write and execute programs.

2) Write the steps to create a C program file in the IDE of your lab computer.

Ans: A software that provides a programming environment to facilitate programmers in writing and executing computer programs is known as an Integrated Development Environment (IDE). An IDE has a graphical user interface (GUI), meaning that a user can interact with it using windows and buttons to provide input and get output. An IDE consists of tools that help a programmer throughout the phases of writing, executing and testing a computer program. This is achieved by combining text editors, compilers and debuggers in a single interface. Some of the many available IDEs for C programming language are:

1. Visual Studio
2. Xcode
3. Code::Bocks

A text editor is a software that allows programmers to write and edit computer programs. All IDEs have their own specific text editors. It is the main screen of an IDE where we can write our programs. Computers only understand and work in machine language consisting of 0s and 1s. They require the conversion of a program written in programming language to machine language in order to execute it. This is achieved using a compiler. A compiler is a software that is responsible for conversion of a computer program written in some high level programming language to machine language code.

3) Describe the purpose of a compiler.

Ans: Computers only understand and work in machine language consisting of 0s and 1s. They require the conversion of a program written in programming language to machine language, in order to execute it. This is achieved using a compiler. A compiler is a software that is responsible for conversion of a computer program written in some high level programming language to machine language code.

4) List down five reserve words in C programming language.

Ans: The five reserved words are as follows

(i) auto (ii) double (iii) int (v) if

C language has total 32 reserved words

Reserved Words in 'C' language

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

5) Discuss the main parts of the structure of a 'C' program.

Ans: **Structure of a 'C' Program**

A 'C' program can be divided into three main parts:

1. **Link section or header section:** While writing programs in 'C' language, we make extensive use of functions that are already defined in the language. But before using the existing functions, we need to include the files where these functions have been defined. These files are called header files. We include these header files in our program by writing the include statements at the top of program.
2. **Main section of a program:** It consists of main ( ) function. Every 'C' program has a main function and it is the starting point of execution.
3. **Body of main () function:** The body of main () is enclosed in the curly braces {}. All the statements inside the curly braces make the body of main function. The statement printf ("Hello World!"); uses a predefined function *printf* to display the statement Hello World! on computer screen.

6) Why do we use comments in programming?

Ans: **Purpose of Comments in C Programs**

Comments are the statements in a program that are ignored by the compiler and do not get executed. Usually comments are written in natural language e.g. in English language in order to provide description of our code.

Types:

1. Single- line comment //
2. Multi- line comment /* */

7) Differentiate between constants and variables.

Ans:

CONSTANTS	VARIABLES
Each language has a basic set of alphabets (character set) that are combined in an allowable manner to form words, and then these words can be used to form sentences. Similarly, in C programming language we have a character set that includes: Alphabets (A, B, ……, Y, Z), (a, b …. y, z) Digits (0 – 9) Special symbols (~ '!@#% ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /) These alphabets, digits and special symbols when combined in an allowable manner, form constants, variables and keywords (also known as reserved words). We have already discussed the concept of reserved words. In the following, we discuss the concept of constants and variables. **Constants:** Constants are the values that cannot be changed by a program e.g. 5, 75.7, 1500 etc. In C language, primarily we have three types of constants: **Integer Constants:** These are the values without a decimal point e.g. 7, 1256, 30100, 55555, -54, -2349 etc. They can be positive or negative. If the value is not preceded by any sign, it is considered as positive. **Real Constants:** These are the values including a decimal point e.g. 3.14, 15.3333, 75.0, -1575.76, -7941.2345 etc. They can also be positive or negative. **Character Constants:** Any Single small case letter, upper case letter, digit, punctuation mark, special symbol enclosed within ' ' is considered a character constant e.g. '5', '7', 'a', 'X', '! ', '; ' etc.	A variable is actually a name given to a memory location, as the data is physically stored inside the computer's memory. The value of a variable can be changed in a program. It means that, in a program, if a variable contains value 5, then later we can give it another value that replaces the value 5. Each variable has a unique name called identifier and has a data type. Data type describes the type of data that can be stored in the variable. C language has different data types such as int, float, and char. The types int, float and char are used to store integer, real and character data respectively. Table 1.2 shows the matching data types in C language, against different types of data. **Type of Data** | **Matching Data Type in C language** | **Sample Values** Integer | int | 123 Real | float | 23.5 Character | char | 'a'

8) Write down the rules for naming variables.

Ans:

1. A variable name can only contain alphabets (uppercase or lowercase), digits and underscore sign.
2. Variable name must begin with a letter or an underscore, it cannot begin with a digit.
3. A reserved word cannot be used as a variable name.
4. There is no strict rule on how long a variable name should be, but we should choose a concise length for variable name to follow good design practice.
   Some examples of valid variable names are height, Average Weight, _var1.

9) Differentiate between char and int.
Ans:

Integer – int (signed/unsigned)	Character-char
Integer data type is used to store integer values (whole numbers). Integer takes up 4 bytes of memory. To declare a variable of type integer, we use the keyword int. **Signed int:** A signed int can store both positive and negative values ranging from -2, 147, 483, 648 to 2,147, 483, 647 By default. Type int is considered as a signed integer. **Unsigned int:** An unsigned into can store only positive values and its value ranges from 0 to +4,292,967,295. Keyword unsigned int is used to declare an unsigned integer.	To declare character type variables in C, we use the keyword char. It takes up just 1 byte of memory for storage. A variable of type char can store one character only.

10) How can we declare and initialize a variable?
Ans:

Variable Declaration	Variable Initialization

We need to declare a variable before we can use it in the program. Declaring a variable includes specifying its data type and giving it a valid name. Following syntax can be followed to declare a variable.
data_type variable name;
Some examples of valid variable declarations are as follows:
unsigned int age;
float height;
int salary;
char marital_status;
Multiple variables of same data type may also be declared in a single statement, as shown in the following examples: unsigned int age, basic_ salary, gross_ salary;
int points_ scored, steps;
float height, marks;
char marital_status, gender;
A variable cannot be declared unless we mention its data type. After declaring a variable, its data type cannot be changed. Declaring a variable specifies the type of variable, the range of values allowed by that variable, and the kind of operations that can be performed on it. Following example shows a program declaring two variables:
EXAMPLE CODE
void main ()
{
char grade;
int value;
}
	Assigning value to a variable for the first time is called variable initialization. C language allows us to initialize a variable both at the time of declaration, and after declaring it. For initializing a variable at the time of declaration, we use the following general structure.
	data_type variable name = value;
	Following example shows a program that demonstrates the declaration and initialization of two variables.
	example code
	#include<stdio.h>
	void main ()
	{
	char grade; //Variable grade is declared
	int value = 25; /*Variable value id declared and initialized. */
	grade = 'A'; //Variable grade is initialized
	}

PROGRAMMING EXERCISE

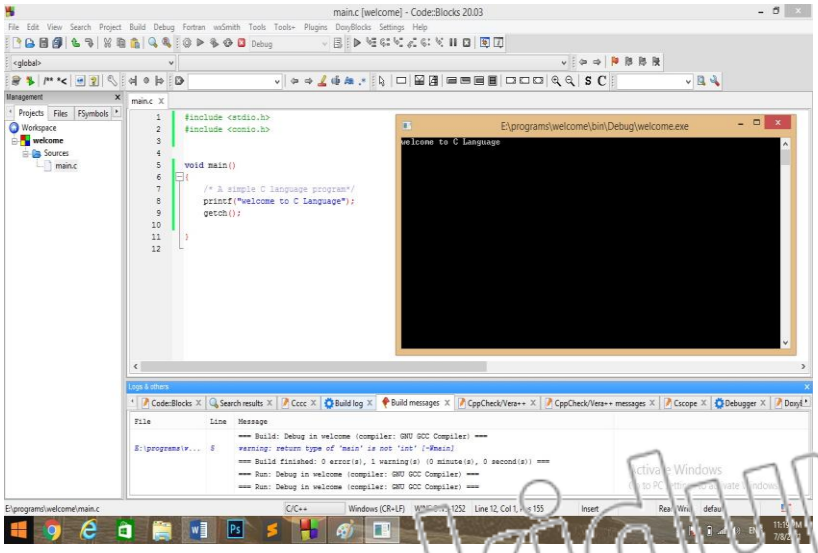
Exercise 1

- **With the help of your teacher open the IDE installed on your lab computer for writing C programs.**
- **Write the following program in the editor and save it as "welcome.c".**

```
#include <stdio.h>
#include <conio.h>
void main ()
{
//A simple C language program
printf ("Welcome to C language");
getch ( );
}
```

Run the program to see Welcome to C language printed on the screen as output.Solution:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  /* A simple C language program*/
   printf("welcome to C Language");
   getch();
}
```



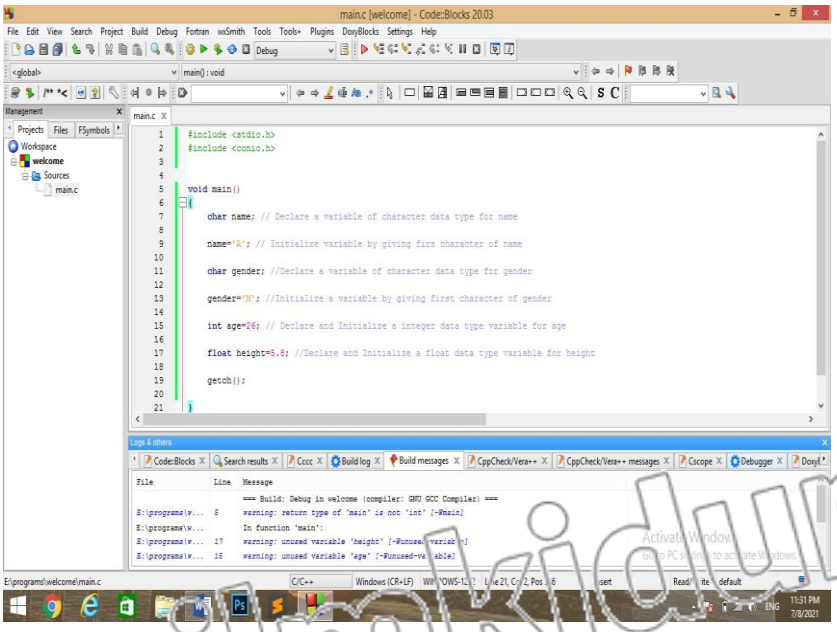
Exercise 2

Write a program that declares variables of appropriate data types to store the personal data about your best friend. Initialize these variables with the following data:

- **Initial letter of his name**
- **Initial letter of his gender**
- **His age**
- **His height**

Solution:

```
#include <stdio.h>
#include <conio.h>
void main()
{
char name; // Declare a variable of character data type for name
name='A'; // Initialize variable by giving firs character of name
char gender; //Declare a variable of character data type for gender
gender='M'; //Initialize a variable by giving first character of gender
int age=26; // Declare and Initialize a integer data type variable for age
float height=5.8; //Declare and Initialize a float data type variable for height
getch();
}
```



ANSWER KEY

1.1 PROGRAMMING EVIRONMENT

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	D	B	A	D	A	A	D	C	D	A	D	A	A	C	A
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
A	D	C	A	A	D	D	D	A	B	A	A	A	A	A	B

1.2 PROGRAMMING BASIC

1	2	3	4
A	A	A	A

1.2.1 RESERVED WORDS

1	2	3	4
D	B	D	C

1.2.2 STRUCTURE OF C PROGRAM

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
D	D	B	D	A	B	D	B	C	C	A	A	A	D	A	A
17	18	19	20	21	22	23									
B	B	A	B	B	B	B									

1.2.3 COMMENTS IN C

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
D	A	C	C	C	B	C	B	C	B	A	C	B	D	A	B

1.3 CONTANTS AND VARIABLE

1.3.1 CONTANTS

1.3.2 VARIABLES

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C	D	B	A	B	C	C	C	A	A	B	D	A	C	D	C
17	18	19	20	21											
B	A	D	A	D											

1.3.3 DATA TYPES OF A VARIABLE

1	2	3	4	5	6	7	8
C	D	D	B	C	C	D	B

1.3.4 NAME OF A VARIABLE

1	2
D	B

1.3.5 VARIALE DECLARATION

1	2	3	4	5
B	C	B	B	B

1.3.6 VARIABLE INTILIZATION

1	2
B	C

CH # 2

USER INTERFACE

2.1 INPUT / OUTPUT (I/O) FUNCTION

2.1.1 PRINTF ()
2.1.2 FORMAT SPECIFIERS
2.1.3 SCANF ()
2.1.4 GETCH ()
2.1.5 STATEMENT TERMINATOR
2.1.6 ESCAPE SEQUENCE

LONG QUESTIONS

1. Explain *printf* ( ) function with some examples. (K.B+U.B+A.B)

Ans. *printf* is a built-in function in 'C' programming language to show output on screen. Its name comes from "***Print formatted***" that is used to print the formatted output on screen. All data types can be displayed with ***printf*** function.

Syntax:

printf (control string, list of arguments) The control string consists of the following:

- Text
- The format specifier
- The escape sequence.

Text is written within double quotes. It specifies the message that is to be printed along with the values. **Format specifier** specifies the format according to which a value is to be printed. **Escape sequence** controls the printing on the output device. **List of arguments** consists of a list of variables or arithmetic expressions, separated by commas, whose values are to be printed. The values are printed according to the corresponding format specifier. The first format specifier applies to the first argument, the second to the second argument and so on. The arguments the ***printf* ( )** function are optional. When ***printf* ( )** function is used to print only a message, then the arguments are omitted.

Example:

```
# include <stdio. h>
void main ()
{
printf ("Hello World");
}
```

Output:

Hello World

In this example, printf function is used to display **Hello World** on computer screen. Whatever we write inside the double quotes (" ") in the *printf* () function, gets displayed on computer screen.

2. Explain format specifier in C language in detail. (K.B+U.B)

Ans: A format specifier is computer code that tells about the data type, field width and the format according to which a value is to be printed or read from an input device. A list of commonly used format specifiers is given below.

- %d (decimal integer)
- %i (integer)
- %1d (long decimal integer)

- %f floating-point (decimal notation)
- %g floating-point (exponential notation)
- %e floating-point (%f or %g, whichever is shorter)
- %c (single character)
- %s (string)

Example:

```
#include < stdio.h>
void main ()
{
float height = 5.8
int age = 35;
printf ("My age is %d and my height is %f", age,height);
}
```

Output:

My age is 35 and my height is 5.800000.We can observe that while displaying output, first format specifier is replaced with the value of first variable/ data after the ending quotation mark i.e ***age*** in the above example, and second format specifier is replaced with the second variable/ data i.e ***height***.

3. Explain *scanf* ( ) function with examples. (K.B+U.B)

Ans:

SCANF ( ) FUNCTION

scanf is a built-in function in 'C' language that takes input from user into the variables. We specify the expected input data type in scanf function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.

Syntax:

scanf ("control/prompt string", list of variables);

The control string specifies the format specifiers. It is written within double quotes. The control string in ***scanf*** **( )** function is different from ***printf*** **( )** function. In scanf **( )** function, strings cannot be given.

We can take multiple inputs using a single scanf function e.g. look at the following statement. scanf ("%d%d%f", &a, &b, &c);

It takes input into two integer type variables a and b, and one float type variable c. After each input, user should enter a "space" or "enter" key. After all the inputs user must press enter key.

Example:

```
#include <stdio.h>
void main ()
{
char grade;
scanf ("%c",&grade);
}
```

In this example, %c format specifier is used to specify character type for the input

variable. Input entered by user is saved in variable grade.

Solution:

```
# include <stdio.h>
void main ()
{
int roll_number;
float percentage;
char grade;
print (“Enter your roll number”);
scanf (“%d, & roll_number);
printf (“Enter percentage of marks”);
printf (“Enter grade”);
scanf (“%c”, & grade);
printf (“Roll no \t: \t %d \n”, roll_number);
printf (“Percentage \t: \t %f%”, percentage);
printf (“Grade \t %c”, grade);
}
```

4. What is *getch* () function? (K.B+A.B)

Ans:

Getch ( ) Function

getch () function is used to read a character from user. The character entered by user does not displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key. To use this function, we need to include the library conio.h in the header section of program.

Example Code

```
# include <stdio.h>
# include <conio.h>
void main ()
}
printf (“Enter any key if you want to exit program”);
getch ()
}
```

5. Explain escape sequence in detail with one program example. (K.B+U.B+A.B)

Ans:

Escape Sequence

The special characters used in ‘C’ language to control printing on the output device are called escape sequences. Escape sequences are used in *printf* function inside the double quotes (“ ”). They force *printf*() to change its normal behavior of showing output. These characters are not printed. These are used inside the control string.

Control Character:

An escape sequence is a combination of a backslash (\) and a code character. The backslash is called the control character. A list of commonly used escape sequences is given below with their meanings.

Commonly used Escape sequences are shown in the table

Sequence	Purpose	Sequence	Purpose

\'	Displays Single Quote	\a	Generates an alert sound
\\	Displays Back slash (\)	\b	Removes previous char
\n	Creates new Line	\t	Moves to next tab

Example:

```
# include <stdio.h>
void main ()
{
printf ("My name is Ali, \n");
printf ("I live in Lahore,");
}
```

Output

My name is Ali
I live in Lahore.

SHORT QUESTIONS

Q.1 What is meant by input and output in programming? (K.B)

Ans:

Input & Output

In programming, input means to enter or feed data in a computer program and output is what the computer produces after processing the data. 'C' language provides many input/output functions to provide interaction between a program and user.

Q.2 Define output function in 'C' language. (K.B)

Ans:

Output Function

"In computer programming, output means to display information on screen or print on printer. 'C' language provides the ***printf*( ), *putchar*( )** and ***puts*( )** functions to output information on computer screen".

Q.3 Name some output function in 'C' language. (K.B)

Ans:

Output Function

Some commonly used output function in C language are:

- printf( )
- putchar( )
- puts( )

Q.4 Define printf( ) function? (K.B)

Ans:

printf( ) Function

printf is a built-in function in 'C' programming language to show output on screen. Its name comes from "Print formatted" that is used to print the formatted output on screen. All data types can be displayed with printf function.

Syntax:

printf ("control string", list of arguments);

Q.5 What is input function in 'C' language? (K.B)

Ans:

scanf Function

In computer programming input means to feed data into program through an input device. 'C' language provides the scanf( ), getch( ), getchar( ) and gets( ) functions to input data.

Q.6 Enlist some input functions in C language. (K.B+U.B+A.B)

Ans: **Input Function**

Some commonly used input functions in 'C' language are:

- scanf( )
- getch( )
- getchar( )
- gets( )

Q.7 What is role of format specifier in 'C' language? (K.B)

Ans: **Format Specifier**

A format specifier is computer code that tells about the data type, field width and the format according to which a value is to be printed or read from an input device

Q.8 Enlist commonly used format specifiers. (K.B)

Ans: **Format Specifiers**

A list of commonly used format specifiers is given below:

- %d decimal integer
- %i integer
- %ld long decimal integer
- %f floating-point (decimal notation)
- %g floating-point (exponential notation)
- %e floating-point (%f of %g, whichever is shorter)
- %c single character
- %s string

Q.9 Differentiate between integer and floating-point format specifier. (K.B+U.B)

Ans: **DIFFERENTIATION**

Following are the differences between integer and floating-point format specifier:

Integer Format Specifier	Floating Point Format Specifier
• The format specifier %d is used to read or print a decimal integer.	• The format specifier %f is used to read and print floating point numbers in decimal notation with a precision of 6 digits after the decimal point.
• The format specifier %ld is used with long integers.	• The format specifier %e is used to read and print floating-point numbers in exponential notation.
• The format specifier %i is used to read or print an integer.	• The format specifier %g is used to print floating-point numbers in decimal or exponential notation whichever is shorter.

Q.10 What is the use of character format specifier? (K.B+U.B)

Ans: **Format Specifier**

The character format specifier, %c is used to read or print a single character.

Q.11 What is *scanf( )* function? (K.B+U.B)

Ans: **Scanf ( ) Function**

scanf is a built in function in C language that takes input from user into the variables. We specify the expected input data type in *scanf* function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.

Syntax:

scanf ("control string", "list of variables");

Q.12 What is *getch*( ) functions? (K.B+U.B)

Ans: **Getch ( ) Functions**

getch () function is used to read a character from user. The character entered by user does not displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key.

Q.13 What is meant by statement terminator in C language? (K.B+U.B)

Ans: **Statement Terminator**

A statement terminator is identifier for compiler which identifies end of a line. In 'C' language semi colon (;) is used as statement terminator. If we do not end each statement with a **(;)** it results into an error/s.

Q.14 Enlist commonly used escape sequence. (K.B+U.B+A.B)

Ans: **Escape Sequence**

Following are the some commonly used escape sequence in 'C' language:

Sequence	Purpose	Sequence	Purpose
\'	Displays Single Quote	\a	Generates an alert sound
\\	Displays Back slash (\)	\b	Removes previous char
\n	Creates new Line	\t	Moves to next tab

Q.15 What is the purpose of escape sequence in 'C' language? (K.B+A.B)

Ans: **Escape Sequence**

The special characters used in 'C' language to control printing on the output device are called escape sequences. Escape sequences are used in ***printf*** function inside the "and." they force ***printf*()** to change its normal behavior of showing output. These characters are not printed. These are used inside the control string.

MUTLIPLE CHOICE QUESTIONS

1. Which operations are involved in communication between computer and user? (K.B)

(A) Input/output Operation (B) Storage Operation
(C) Processing Operation (D) Controlling Operation

2. 'C' language uses function to provide interaction between program and user: (K.B)

(A) Input Function (B) Output Function (C) Format Specifier (D) Both A & B

3. Which one of the following is not a output function: (K.B+U.B)

(A) printf( ) (B) putchar ( ) (C) puts ( ) (D) getch( )

4. In 'C' program to display text, variable, constant and expression we use: (K.B)

(A) printf( ) (B) puts( ) (C) putchar( ) (D) strcpy( )

5. Which function is used to get values into variables from the keyboard during the execution of a program: (K.B+U.B)

(A) scanf( ) (B) getche( ) (C) printf( ) (D) getchec( )

6. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main()
{
printf("Hello World! %d \n", x);
return 0;
}
```

(A) Hello World! x; (B) Hello World! followed by a Junk Value
(C) Compile Time Error (D) Hello World!

7. The format identifier '%i' is also used for data type: (K.B)

(A) char (B) int (C) float (D) double

8. Find the output of the following program. (A.B)

```
void main()
{
int i=065, j=65;
printf("%d %d", i, j);
}
```

(A) 53 65 (B) 65 65 (C) 065 65 (D) 053 65

9. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int var = 010;
printf("%d", var);
}
```

(A.B)

(A) 2 (B) 8 (C) 9 (D) 10

10. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
printf("sanfoundary\rclass\n");
return 0;
}
```

(A) sanfoundaryclass (B) sanndry (C) sanfoundarclass (D) sanfoundary

11. *scanf*( ) returns as its value:

(A) Number of Successfully Matched and Assigned Input Items
(B) Nothing
(C) Number of Characters Properly Printed
(D) Error

12. '%f' access specifier is used for: (A.B)

(A) Strings (B) Integral Types (C) Floating Types (D) Character Types

13. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
printf("%.0f",2.89);
```

(A) 2.890000 (B) 2.89 (C) 2 (D) 3

14. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h> int main( )
{
float a = 2.455555555555
printf("%f", a);
}
```

(A) 2.455555 (B) 2.455556 (C) 2.456 (D) 2.46

15. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int i = -3;
int k = i % 2'
printf("%d\n", k);
}
```

(A) Compile Time Error (B) -1
(C) 1 (D) Implementation Defined

16. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int i = 5;
i = i / 3;
printf("%d/n", i);
return 0;
}
```

(A) Compile Time Error (B) 3
(C) 1 (D) Implementation Defined

17. A symbol used in 'C' with *scanf*( ) function: (K.B+A.B)
(A) # (B)! (C) & (D) i

18. Which one of the following symbol is used as a statement terminator? (K.B)
(A) : (B) # (C) ; (D) }

19. A computer code that tells about data type field width and function is called _____.(K.B)
(A) Reserve Word (B) Format Specifier (C) Escape Sequence (D) String Field

20. Which format specifier is used for decimal integer data? (K.B+A.B)
(A) % (B) % Id (C) %d (D) %C

21. A format specifier use for integer_____. (K.B+A.B)
(A) %d (B) i% (C) %f (D) %c

22. A format specifier use for long decimal integer _____. (K.B+A.B)
(A) %d (B) %i (C) %id (D) %f

23. Format specifier used for string variable _____. (K.B+A.B)
(A) %s (B) %c (C) %e (D) %g

24. A format specifier used for floating point exponential notation _____. (K.B+A.B)
(A) %e (B) %g (C) %c (D) %f

25. A format specifier is used for single character _____. (K.B+A.B)
(A) %c (B) %s (C) %e (D) %

26. The special character used in C language to control printing on output devices are called _____ (K.B+A.B)
(A) Format Specifier (B) Escape (C) String Format (D) Control String

27. Which Escape sequence is used to produce a test (bell) sound? (K.B+A.B)
(A) \a (B) \b (C) \n (D) \r

28. A escape sequence used to move cursor back word by one position. _____. (K.B+A.B)
(A) \b (B) \n (C) \r (D) \t

29. Which escape sequence is used to move cursor to the beginning? (K.B+A.B)

(A) \a (B) \b (C) \n (D) \r

30. A escape sequence used to move cursor to the next horizontal tabular position:(K.B+A.B)
(A) \t (B) \\ (C) \n (D) \

31. Which escape sequence produced a back slash? (K.B+A.B)
(A) \ (B) \\ (C) \' (D) \"

32. Escape sequences are prefixed with _____. (K.B+A.B)
(A) % (B) \ (C) \\ (D) \"

33. Format specifiers are used for _____. (K.B+A.B)
(A) Variable (B) Constants
(C) Both (D) All of these

34. We can take_____ inputs using single scan function. (K.B+A.B)
(A) Single (B) Multiple (C) Ten (D) None of these

35. If we forget '&' symbol in scanf function it is _____. (K.B+A.B)
(A) Error (B) Not an error (C) Code (D) None of these

36. Escape sequence used for new line _____. (K.B+A.B)
(A) \a (B) \b (C) \n (D) None of these

37. Escape sequence to display a single quote _____. (K.B+A.B)
(A) \a (B) \' (C) \, (D);

38. A tab is a collection of _____ spaces. (K.B+A.B)
(A) 8 (B) 12 (C) 16 (D) 32

2.2 OPERATORS

LONG QUESTIONS

1. What is the use of arithmetic operators? Also enlist its types. (K.B+A.B)

Ans:

ARITHMETIC OPERATORS

Arithmetic operators are used to perform arithmetic operations that include addition, subtraction, multiplication, division and also to find the remainder obtained when an integer is divided by another integer.

Types:

The types of arithmetic operators used in C language are described here:

Operator	Name	Description
/	Division Operator	It is used to divide the value on left side by the value on right side.
*	Multiplication Operator	It is used to multiply two values.
+	Addition Operator	It is used to add two values.
-	Subtraction Operator	It is used to subtract the value on right side from the value on left side.
%	Modulus Operator	It gives remainder value after dividing the left operand by right operand.

Program

```
/* This program takes an input the price of a box of chocolates and the total number of chocolates in the box. The program finds and displays the price of one chocolate */
#include <stdio.h>
void main ()
{
```

```
float box_price, num_of_chocolates, unit_price;
print ("Please enter the price of whole box of chocolates:");
scanf ("%", & box_price);
printf ("Please enter the number of chocolates in the box:");
scanf ("&f", &num_of_chocolates);
unit_price = box_price / num_of_chocolates;
printf ("The price of a single chocolate is %f", unit_price);
}
```

Output:

Please enter the price of whole box of chocolates: 150
Please enter the number of chocolates in the box: 50
The price of a single chocolate is 3.000000

2. Explain relational operators in detail. (K.B+U.B)

Ans:

RELATIONAL OPERATORS

Relational operators compare two values to determine the relationship between values. Relational operators identify either the values are equal, not equal, greater than or less than one another. 'C' language allows us to perform relational operators on numeric and char type data. Table presents relational operators in C language and their descriptions:

Relational Operator	Description
= =	Equal to
! =	Not equal
>	Greater than
<	Less than
> =	Greater than equal to
< =	Less than equal to

Relation operators perform operations on two operands and return the result in Boolean expression (***TRUE OR FALSE***). A true value is represented by 1, whereas a false value is represented by a 0. This concept is further illustrated as followed

Examples:

Relation Expression	Explanation	Result
5 = =5	5 is equal to 5 ?	TRUE
5 ! = 7	5 is not equal to 7?	TRUE
5 > = 5	5 is greater than or equal to 5?	TRUE
5 < = 4	5 is less than or equal to 4?	FALSE

3. Explain Logical operators in detail (K.B+U.B)

LOGICAL OPERATORS

Logical operators are used for building compound conditions. We have seen before that a single condition is built using a relational operator in an expression. If we need to build more than one condition for some action to take place in programming, then we have to form compound condition.

Types of Logical Operators:

There are three types of logical operators. These are:

Operator	Definition
& &	AND
||	OR
!	NOT

AND Operator (&&):

AND operator && takes two Boolean expressions as operands and produces the result TRUE if both of its operands are TRUE. It returns FALSE if any of the operands is FALSE. Table shows the truth table for AND operator.

Expression	Result
FALSE && FALSE	FALSE
FALSE && TRUE	FALSE
TRUE && FALSE	FALSE
TRUE && TRUE	TRUE

OR Operator (||):

OR operator accepts Boolean expression and returns TRUE if at least one of the operands it TRUE. Table shows that truth table for OR operator.

Expression	Result
FALSE || FALSE	FALSE
FALSE || TRUE	TRUE
TRUE || FALSE	TRUE
TRUE || TRUE	TRUE

NOT Operator (!):

NOT operator negates or reverses the value of Boolean expression. It makes it TRUE, if it is FALSE and FALSE if it is TRUE. Table present the truth table for Not operator.

EXPRESSION	Result
!(TRUE)	FALSE
!(FALSE)	TRUE

Examples of Logical Operators:

Logical Expression	Explanation	Result
3 < 4 && 7 > 8	3 is less than 4 AND 7 is greater than 8?	FALSE
3 = = 4 || 3 > 1	3 is equal to 4 OR 3 is greater than 1?	TRUE
!(4 > 2 || 2 = = 2)	NOT (4 is greater than 2 OR 2 is equal to 2)?	FALSE

Example:

The expression:

!(a<b)

Will be true if **a** is not less than **b**. In other words, the condition will be true if a is greater than or equal to **b**. The same condition can also be written as **(a>=b)** which is easy to understand.

SHORT QUESTIONS

Q.1 What is an expression? (K.B)

Ans:

EXPRESSION

Expressions consist of constants and variables combined together with operators.

Example:

Percentage is = obtained_marks / total_marks*100;

Q.2 Define operators in 'C' language.

Ans: **OPERATORS IN 'C' LANGUAGE**

"An operator is a symbol used to command the computer to perform a certain mathematical or logical operation. Operators are used to operate on data and variables".

Types:

Some commonly used operators in 'C' language are:

- Arithmetic operators
- Assignment operators
- Relational operators
- Logical operators

Q.3 Name some operators of 'C' language. (K.B)

Ans: **'C' LANGUAGE OPERATORS**

Following are some commonly used operators in C language:

- Arithmetic operators
- Assignment operators
- Relational operators
- Logical operators

Q.4 What are arithmetic operators in 'C' language? (K.B)

Ans: **ARITHMETIC OPERATORS**

Arithmetic operators are used to perform arithmetic operations that include addition, subtraction, multiplication, division and also to find the remainder obtained when an integer is divided by another integer.

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder (Modulus) Operator

Q.5 What is Modulus Operator? (K.B)

Ans: **MODULUS OPERATOR**

Modulus operator (%) performs division of left operand by the right operand and returns the remainder value after division. Modulus operator works on integer data types.

- **int REM = 14 % 3;**

As, when we divide 14 by 3, we get a remainder of 2, so the value stored in variable REM is 2.

Q.6 What are logical operators in 'C' language? (K.B)

Ans: **LOGICAL OPERATORS**

Logical operators are used for building compound conditions. We have seen before that a single condition is built using a relational operator in an expression. If we need to build more than one condition for some action to take place in programming, then we have to form compound condition.

Types:

There are three logical operators:

- AND (&&)
- OR (||)
- NOT (!)

Q.7 Differentiate between relational and logical operators. (K.B+U.B)

Ans: **DIFFERENTIATION**

Following are the differences between relational and logical operators:

Relational Operator	Logical Operator
• Relational operators are used to compare two values of the same types.	• Logical operators are used for building compound conditions.
• After evaluation of a relational expression, the result produced is either True or False.	• These operator always evaluates results in the form of true and false.
• Relational operator include ==, !=, <, >, <= and >=.	• Logical operators include, AND (&&), OR (||), and NOT (!).

Q.8 Enlist name of Relational operators. **(K.B)**

Ans: **RELATIONAL OPERATORS**

Relational Operator	Description
= =	Equal to
! =	Not equal
>	Greater than
<	Less than
> =	Greater than equal to
< =	Less than equal to

Q.9 Enlist name of logical operators. **(K.B)**

Ans: **LOGICAL OPERATORS**

There are three types of logical operators. These are described below.

Operator	Definition
& &	AND
||	OR
!	NOT

Q.10 Draw Table for AND Operator. **(K.B)**

Ans: **AND OPERATOR**

Expression	Result
FALSE && FALSE	FALSE
FALSE && TRUE	FALSE
TRUE && FALSE	FALSE
TRUE && TRUE	TRUE

Q.11 Draw Table for OR Operator. **(K.B)**

Ans: **OR OPERATORS**

Expression	Result
FALSE || FALSE	FALSE
FALSE || TRUE	TRUE
TRUE || FALSE	TRUE
TRUE || TRUE	TRUE

Q.12 Draw Table for NOT Operator. **(K.B)**

Ans: **NOT OPERATORS**

Expression	Result
!(TRUE)	FALSE
!(FALSE)	TRUE

Q.13 Write down the differences between Assignment operator (=) and Equal to operators (= =)? (K.B+U.B)

Ans: Following are the differences between assignment operator and equal to operator.

Assignment Operator (=)	Equal to Operators (= =)
Assignment operator is used to assign a value to a variable, or assign a value of variable to another variable.	In C language, = = operator is used to check for equality of two expressions
Equal sign (=) is used as assignment operator in C.	Double Equal sign (= =) is used as equal to operator
Consider the following example: int sum = 5;	Consider the following example: 3+2 = = 5;

Q.14 Differentiate between Unary and Binary operators. (K.B+U.B)

Ans:

Unary Operators:	Binary Operators:
Unary operators are applied over one operand only	Binary operators require two operands to perform the operation
Example Logical not (!) operator has only one operand. Sign operator (-) is another example of a unary operator e.g. -5.	**Example** All the arithmetic operators, and relational operators are binary operators. The logical operator && and || are also binary operators.

Q.15 What do you know about operator precedence in 'C' language? (K.B+U.B)

Ans: **ORDER OF PRECEDENCE OF OPERATORS**

If there are multiple operators in an expression, the question arises that which operator is evaluated first. To solve this issue, a precedence has been given to each operator. An operator with higher precedence is evaluated before the operator with lower precedence. In case of equal precedence, the operator at left side is evaluated before the operator at right side.

Example:

Result = 18 / 2 * 3 + 7% 3 + (5 * 4);

Operator	Precedence
()	1
!	2
*, /, %	3
+, -	4
>, <, >=, <=	5
==, !=	6
&&	7
||	8
=	9

MULTIPLE CHOICE QUESTIONS

1. Which of the following is related to expression? (K.B)
(A) Control / Variable (B) Operators
(C) Mathematical Operation (D) All of these

2. 'C' language has ___ main types of operators. (K.B)
(A) 3 (B) 4 (C) 5 (D) 6

3. In 'C' language we used to program arithmetic operations by using _____. (K.B)
(A) Arithmetic Operators (B) Assignment
(C) Relational Operator (D) Logical Operator

4. Assignment operators is 'C' language is denoted by _____. (K.B+U.B)
(A) = (B) == (C) <=i (D) =>

5. In relational operators equal to is denoted by: (K.B+U.B)
(A) == (B) = (C)! = (D) >=

6. Which operators are used to build compound condition? (K.B+U.B)
(A) Assignment (B) Relational (C) Logical (D) Increment

7. In 'C' language "AND" logical operator is denoted by using symbol _____.(K.B+U.B)
(A) && (B) // (C)! (D) #

8. In 'C' language "OR" logical operator is denoted by _____. (K.B+U.B)
(A) && (B) || (C) # (D)!

9. Which logical operator uses single expression? (K.B+U.B)
(A) AND (B) OR (C) NOT (D) NAND

10. Expression !(a<b) is an example of _____. (K.B+U.B)
(A) AND (B) OR (C) NOT (D) XOR

11. Which one operator has highest order of precedence in 'C'? (K.B+U.B)
(A) ++, -- (B) *, / % (C) +, - (D) ||

12. Which one of the following operators of C has lowest order of precedence?(K.B+U.B)
(A) Logical Operator (B) Assignment
(C) Relational Operators (D) Increment / Decrement Operator

13. What will be the output value of x in the following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int x = 5 * 9 / 3 + 9;
}
```

(A) 3.75 (B) Depends on Compiler
(C) 24 (D) 3

14. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int x = 5.3 % 2;
printf("value of x is %d", x);
```

(A) Value of x is 2.3 (B) Value of x is 1
(C) Value of x is 0.3 (D) Compile Time Error

15. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int y = 3;
int x = 5 % 2 * 3 / 2;
printf(:Value of x is %d:, x);
}
```

(A) Value of x is 1 (B) Value of x is 2
(C) Value of x is 3 (D) Compile Time Error

16. The precedence of arithmetic operators is (from Highest to Lowest): (K.B+U.B+A.B)
(A) %, *, /, +, - (B) %, +, /, *, - (C) +, -, %, *, / (D) %, +, -, *, /

17. Which one of the following is not an arithmetic operation? (K.B+U.B+A.B)
(A) a*=10; (B) a/=10; (C) a != 10; (D) a%=10;

18. Which one of the following data type will throw an error on modulus operation (%)?(K.B+U.B+A.B)
(A) Char (B) Short (C) int (D) float

19. Which one of the following are the fundamental arithmetic operators?(K.B+U.B+A.B)
(A) +, - (B) +,-,% (C) +,-,*,/ (D) +,-,*,/,%

20. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int a = 10;
double b = 5.6;
int c;
c = a + b;
printf("%d", c);
}
```

(A) 15 (B) 16 (C) 15.6 (D) 10

21. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int a = 10, b = 5 , c = 5;
int d;
d = a == (b+c);
printf("%d", d);
```

(A) 1 (B) 5 (C) 5 (D) 25

22. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int x = 1, y= 0, z=5;
int a = x && y || z++;
printf("%d", z);
}
```

(A) 6 (B) 5 (C) 0 (D) Varies

23. What will be the output of following 'C' code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int x = 1, z= 3;
int y = x << 3;
printf("%d\n", y);
}
```

(A) -2147483648 (B) -1 (C) 1 (D) Runtime Error

24. Result of a logical or relational expression in C language is: (K.B+U.B+A.B)

(A) True or False

(B) 0 or 1

(C) 0 If Expression is False and any Positive Number if Expression is True

(D) Both a & b

25. What will be the value of d in the following C program? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int a = 10,   b = 5 , c = 5;
int d
d = b + c == a;
printf("%d", d);
}
```

(A) 1 (B) 10 (C) 10 (D) 25

26. Which among the following is NOT a logical or relational operator? (K.B+U.B+A.B)

(A)! = (B) == (C) || (D) =

27. Relational operators cannot be used on: (K.B+U.B+A.B)

(A) Structure (B) Long (C) String (D) float

28. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int x = 2,    y = 0;
int z = (y++) ? 2 : y == 1 && x ;
printf("%d\n", z);
return 0;
}
```

(A) 0 (B) 1 (C) 2 (D) Undefined Behavior

29. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
int main( )
{
int y = 2;
int z = y + (y = 10);
printf("%d\n", z);
}
```

(A) 12 (B) 20 (C) 4 (D) Either 12 or 20

30. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int b = 5 & 4 & 6;
printf("%d", b);
}
```

(A) 3 (B) 4 (C) 5 (D) 6

31. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int b = 5 & 4 | 6;
printf("%d", b);
}
```

(A) 0 (B) 4 (C) 1 (D) 6

32. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int b = 5 + 7 * 4- 9 * (3, 2);
printf("%d", b);
}
```

(A) 6 (B) 13 (C) 15 (D) 21

33. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int h = 8;
int b = 4 * 6 + 3 * 4 < 3 ? 4 : 3;
printf("%d\n", b);
}
```

(A) 3 (B) 33 (C) 24 (D) 11

34. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
int a = 2 + 3 - 4 + 8 – 5 % 4;
printf("%d\n", a);
}
```

(A) 0 (B) 8 (C) 9 (D) 11

35. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
chr  a = '0';
chr  b = 'm';
int c = a && b | | '1';
printf("%d\n", c);
}
```

(A) 0 (B) 1 (C) a (D) m

36. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main( )
{
chr  a = 'A';
chr  b = 'B';
int c = a + b  % 3 - * 2;
printf("%d\n", c);
}
```

(A) 58 (B) 59 (C) 64 (D) 65

37. The variable on which the operations performed are called _____. (K.B+U.B)
(A) Product (B) Expression (C) Operands (D) Operators

38. _____ operators gives the remainder. (K.B+U.B)
(A) Division (B) Modulus (C) Integral (D) None of these

39. If both operands are type the remainder is _____ to give answer in integer. (K.B+U.B)
(A) Repeated (B) Truncated (C) Both A & B (D) None of these

40. The symbol for Modulus operator in C is _____. (K.B+U.B)
(A) * (B) % (C) ! (D) Wood

41. The symbol for multiplication in C is _____. (K.B+U.B)
(A) X (B) x (C) * (D) None of these

42. Relational operators are used to perform operations on _____. (K.B+U.B)
(A) Numeric (B) Characters (C) Strings (D) Both A & B

43. _____ can also be used to show result of relational expression. (K.B+U.B)
(A) printf( ) (B) scanf ( ) (C) Both A & B (D) None of these

44. The symbol used for 'Equal To' operators is _____. (K.B+U.B+A.B)
(A) = (B) == (C) Both A & B (D) None of these

45. The symbol used for 'Not Equal To' operator is _____. (K.B+U.B+A.B)
(A)!= (B) \= (C) ≠ (D) None of these

46. In AND operator's expression FALSE & FALSE results. (K.B+U.B+A.B)
(A) True (B) False (C) Can be both (D) None of these

47. FALS || FALSE always return ____ in OR operator. (K.B+U.B)
(A) False (B) True
(C) can be true or false (D) None

48. ____ is also known as invertor. (K.B)
(A) AND (B) NOT (C) OR (D) None of these

49. The operator which is applied on single operands called _____. (K.B)
(A) Unity Operator (B) Unary operator (C) Binary (D) None of these

50. ____ operator require three operands. (K.B)

(A) Unary (B) Binary (C) Ternary (D) None of these

51. ____ is an example of unary operators (K.B)

(A)! not (B) - (C) Both A & B (D) None of these

52. In AND operator FALSE && TRUE results (K.B+U.B+A.B)

(A) TRUE (B) FALSE

(C) Can be True or False (D) None of these

53. In AND operator the results in only True when both inputs are (K.B+U.B+A.B)

(A) TRUE (B) FALSE

(C) One TRUE one FALSE (D) None of these

54. In OR operator TRUE || FALSE results in _____. (K.B+U.B+A.B)

(A) TRUE (B) FALSE

(C) Can be True or False (D) None of these

55. In OR operator the answer will be FALSE only if both inputs are _____.(K.B+U.B+A.B)

(A) TRUE (B) FALSE (C) TRUE and FALSE (D) None of these

56. Predict the output !(TRUE). (K.B+U.B+A.B)

(A) TRUE (B) FALSE (C) Can be both (D) None of these

57. The variable on which the operations performed are called _____. (K.B+U.B+A.B)

(A) Product (B) Expression (C) Operands (D) Operators

58. Operators are used to perform operations on their _____. (K.B)

(A) Operators (B) Operands (C) Expression (D) None of these

59. _____ operators Gives the remainder. (K.B)

(A) Division (B) Modulus (C) Integral (D) None of these

60. If both the operands of type int, then result of division is also of type integer. Reminder is _____ to give the integer answer. (K.B)

(A) Repeated (B) Truncated (C) Both (D) None of these

61. The symbol used for multiplication in C is _____. (K.B)

(A) X (B) x (C) * (D) None of these

62. The symbol used for Modulus operator in C is _____. (K.B)

(A) * (B) % (C) ! (D) Wood

63. Relational operators are used to perform operations on _____. (K.B)

(A) Numeric (B) Characters (C) Strings (D) Both A & B

PROGRAMMING TIME (A.B)

Programming Time 2.1

Write a program that swaps the values of two integer variables.

Program:

```
void main ()
{
        int a = 2, b = 3, temp;
        temp = a;
        a = b;
        b = temp;
        printf ("Value of a after swapping: %d\n", a);
        printf ("Value of b after swapping: %d\n",  b);
}
```

Programming Time 2.2

/*This program takes as input the price of a box of chocolates and the total number of chocolates in the box. The program finds and displays the price of one chocolate.*/

```
# include <stdio.h>
Void main ( )
{
        float box_price, num_of_chocolates, unit_price;
        printf ("Please enter the price of whole box of chocolates:");
        scanf("%f", &box_price);
        printf ("Please enter the number of chocolates in the box: ");
        scanf("%f, &num_of_chocolates);
        unit_price = box_price / num_of_chocolates;
        printf("The price of a single chocolate is %f, unit_price);
}
```

Output:

```
Please enter the price of whole box of chocolates: 150
Please enter the number of chocolates in the box: 50
The price of a single chocolate is 3.000000
```

Programming Time 2.3

/* Following program takes as input the length and width of a rectangle. Program calculates and displays the area of rectangle on screen. */

```
# include<stdio.h>
void main ()
{
        float length, width, area;
        printf ("Please enter the length of rectangle:");
        scanf ("%f", &length);
        printf ("Please enter the width of rectangle: ");
        scanf ("%f", &width);
        area = length *width;
        printf ("Area of rectangle is : %f", area);
}
```

Output

```
Please enter the length of rectangle: 6.5
Please enter the length of rectangle: 3
Area of rectangle is : 19.500000
```

Programming Time 2.4

```
/* This program takes marks of two subjects from user and displays the sum of marks on console. */
#include <stdio.h>
void main ()
{
        int sum, math, science;
        printf ("Enter marks of Mathematics:");
        scanf ("%d", &math);
        printf ("Enter marks of  Science: ");
        scanf ("%d, &science);
        sum = math + science;
        printf ("Sum of marks is : %d", sum);
}
```

Output

```
Enter marks of Mathematics: 90
Enter marks of Science: 80
Sum of marks is: 170
```

Programming Time 2.5

```
/* This program finds and displays the right most digit of an input number. */
#include <stdio.h>
void  main ()
{
        int num, digit;
        printf("Enter a number:"); scanf("%d, &num);
        digit = num % 10;
        printf("Right most digit of number you entered is: %d", digit);
}
```

Output

```
Enter a number: 789
Right most digit of number you entered is: 9
```

ACTIVITY 2.1

Write down the output of following code:

```
#include <stdio.h>
void main ()
{
   printf ("I am UPPERCASE and this is lowercase");
}
```

Output: I am UPPERCASE and this is lowercase

ACTIVITY 2.2

Write a program that shows your first name in Uppercase and you last name in lower case letters on screen

Solution:

```
#include <stdio.h>
void main()
{
   printf("IMRAN khan");
}
```

Output: IMRAN khan

ACTIVITY 2.3

Write a program that takes roll number, percentage of marks and grade form user as input. Program should display the formatted output like following:

Roll No : input value
Percentage : input value%
Grade : input value

ACTIVITY 2.4

Write a program that takes as input the length of one side of a square and calculates the area of square.

Solution:

```
# include <stdio.h>
void main ()
{
   int Area, Side;
   printf ("enter the length of one side = ");
   scanf ("%d", & Side);
   Area = Side * Side; // calculate area of square
   printf ("Area of square = %d" , Area);
}
```

ACTIVITY 2. 5

Write a program that takes as input the number of balls in jar A and the number of balls in jar B. The program calculates and displays the total number of balls.

Solution:

```
# include <stdio.h>
void main ()
{
   int jar_A, jar_B, Sum;
   printf ("Enter number of balls in Jar A = ");
   scanf ("%d", &jar_A);
   printf ("Enter number of ball in Jar B = ");
   scanf ("%d", & jar_B);
   Sum = jar_A + jar_B;// calculate sum Write print (“sum of jar A and jar B= %d”,sum);
         printf("Sum of jar A and jar B = %d" , Sum);
}
```

ACTIVITY 2.6

Write a program should display the original price of shirt, discount on price and price after discount.

Solution:

```
# include <stdio.h>
void main ()
{
   int original_prince, price, discount_pre; float discounted_price;
   printf ("Enter the original price of shirt ");
   scanf ("%d", &original_prince);
   printf ("enter the discount percentage ");
   scanf ("%d", &discount_pre);
   discounted_price = original_prince * discount_pre /100;
   printf ("original price of shirt = %d discount on price %d and price after discount = %d",
original_prince, discount_pre, discounted_price);
}
```

ACTIVITY 2.7

Write a program that takes 2 digit number from user, computers the product of both digits and show the output.

Solution:

```
# include <stdio.h>
void main ()
{
   int num, rem, prod =1;
   printf ("Enter a number " );
   scanf ("%d" , &num);
   while (num!=0)
   {
      rem = num % 10; // get the right most digit
      prod = prod * rem, // calculate product of digits
      num = num /10; // remove the right most digit
   }
   printf ("%d", prod);
}
```

ACTIVITY 2.8

Write a program that takes second as input and calculates equivalent number of hours, minutes and seconds.

Solution:

```
#include <stdio.h>
void main()
{
   int sec, hour, min, s;
   printf ("input seconds: ");
   scanf ("%d", & sec);
   hour = sec/3600;
   min = (sec-/3600 * hour)160;
   s= (sec - (3600*hour) - (min *60))
   printf ("hour: min:s-;%d: %d:%d\n", hour, min, s);
}
```

ACTIVITY 2.9

Convert the following algebraic expressions into C expressions.

$x = 6y + z$ $\rightarrow x = 6 * y + z$

$x = yz^3 + 3y$ $\rightarrow x = y * z * z * + 3 * y$

$z = x + \frac{y^2}{3x}$ $\rightarrow z = x + \rightarrow y * y / 3 * x$

$z = (x-2)^2 + 3y$ $\rightarrow x * x - 2 * x * 2 + 2 * 2 + 3 * y$

$y = \left(x + \frac{3z}{2}\right) + z^3 + \frac{x}{x}$ $\rightarrow \left(x + (3 * z / 2)\right) + z * z * z + x / z$

ACTIVITY 2.10

Consider the variable x = 3, y =7. Find out the Boolean result of following expression.

(2 + 5) > y	(x + 4) == y
x! = (y - 4)	(y / 2) > = x
-1 < x	(x * 3) < = 20

ACTIVITY 2.11

Assume the following variable values x 4, y = 7 z = 8. Find out the resultant expression.

x = = 2 || y = = 8	7 > = y && z < 5
z > = 5 || x < = -3	y = = y && .(true)
x! = y || y < 5	!(z > x)

ACTIVITY 2.12

Find out the results of the following expression:

Expression	Result
10 / (5 + 3)	2
7 + 3 * (12 + 12)	79
25 %3 * 4	4
34 – 9 * 2 / (3 * 3)	32
18 / (15 – 3 * 2)	2

EXERCISE

Q1. Multiple Choice Questions.

1) 'printf' is used to print _____ type of data. **(U.B+A.B)**
A) *int* B) *float* C) *char* D) All of these

2) 'scanf' is a _____ in C programming language. **(U.B+A.B)**
A) Keyword B) Library C) Function D) None of these

3) getch() is used to take _____ as input from user. **(U.B+A.B)**
A) *int* B) *float* C) *char* D) All of these

4) Let the following part of code, what will be the value of variable '**a**' after execution: **(U.B+A.B)**
int a = 4;
float b = 2.2;
a = a * b;
A) 8.8 B) 8 C) 8.0 D) 8.2

5) Which one of the following is a valid line of code. **(U.B+A.B)**
A) int = 20; B) grade = 'A';
C) line = this is a line; D) none of these

6) Which operator has highest precedence among the following: **(K.B+U.B)**
A) / B) = C) > D) !

7) Which one of the following is not a type of operator: **(K.B+U.B)**
A) Arithmetic operator B) Relational operator
C) Check operator D) Logical operator

8) The operator % is used to calculate _____. **(K.B+U.B)**
A) Percentage B) Remainder C) Factorial D) Square

9) Which one of the following is a valid character: **(K.B+U.B)**
A) here B) "a" C) '9' D) None of these

10) What is true about C language: **(K.B+U.B)**
A) C is not a case sensitive language
B) Keywords can be used as variable names
C) All logical operators are binary operators
D) None of them

ANSWER KEY

1	2	3	4	5	6	7	8	9	10
D	C	C	B	B	D	B	B	C	D

Q2. True or False

1) Maximum value that can be stored by an integer is 32000. **(K.B) T**/F
2) Format specifiers begin with a '%' sign. **(K.B) T**/F
3) Precedence of division operator is greater than multiplication operator. **(K.B)** T/**F**
4) *getch* is used to take all types of data input from user. **(K.B+A.B)** T/**F**
5) scanf is used for output operations. **(K.B+A.B) T**/F

Q3. Define the following. **(K.B)**

1) **Statement Terminator**

Ans: **STATEMENT TERMINATOR**

A statement terminator is an identifier for compiler which identifies end of a line. In C language semicolon (;) is used as statement terminator. If we do not end each statement with a statement terminator it result into error. **Error:** **Missing statement**

2) Format Specifier

Ans:

FORMAT SPECIFIER

A format specifier is computer code that tells about the data type, field width and the format according to which a value is to be printed or read from an input device

A list of commonly used format specifiers is given below:

- %d decimal integer
- %i integer
- %ld long decimal integer
- %f floating-point (decimal notation)
- %g floating-point (exponential notation)
- %e floating-point (%f of %g, whichever is shorter)
- %c single character
- %s string

3) Escape Sequence

Ans:

ESCAPE SEQUENCE

The special characters used in C language to control printing on the output device are called escape sequences. Escape sequences are used in print function inside the "and." they force printf() to change its normal behavior of showing output. These characters are not printed. These are used inside the control string.

4) *scanf*()

Ans:

SCANF () FUNCTION

scanf ()is a built-in function in C language that takes input from user into the variables. We specify the expected input data type in scanf function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.

Syntax:

scanf ("control string", list of variables);

5) Modulus Operator

Ans:

MODULUS OPERATOR

Modulus operator (%) performs division of left operand by the right operand and returns the remainder value after division. Modulus operator works on integer data types.

- **int REM = 14 % 3;**

As, when we divide 14 by 3, we get a remainder of 2, so the value stored in variable REM is 2

Q4. Briefly answer the following questions. (U.3)

1) What is the difference between *scanf* and *getch?*

Ans:

scanf( ) function	getch( ) FUNCTIONS
scanf is a built-in function in C language that takes input from user into the variables. We specify the expected input data type in *scanf* function with the help of format specifier. If user enters integer data type, format specifier mentioned in scanf must be %d or %i.	*getch*() function is used to read a character from user. The character entered by user does not at displayed on screen. This function is generally used to hold the execution of program because the program does not continue further until the user types a key.
Syntax: *scanf*("control string", list of variables seperrated with commas);	**Syntax:** getch();

2) Which function of C language is used to display output on screen?

Ans: **PRINTF ( ) FUNCTION**

printf is built-in function in C programming language to show output on screen. Its name comes from "Print formatted" that is used to print the formatted output on screen. All data types can be displayed with *printf* function.

Syntax:

printf ("control string", list of arguments);

3) Why format specifiers are important to be specified in I/O operations?

Ans: **FORMAT SPECIFIER**

A format specifier is computer code that tells about the data type, field width and the format according to which a value is to be printed or read from an input device

Example:

- %d decimal integer
- %i integer
- %ld long decimal integer
- %f floating-point
- %s string

4) What are escape sequences? Why do we need them?

Ans: **ESCAPE SEQUENCE**

The special characters used in C language to control printing on the output device are called escape sequences. Escape sequences are used in print function inside the "and." they force printf() to change its normal behavior of showing output. These characters are not printed. These are used inside the control string.

5) Which operators are used for arithmetic operations?

Ans: **ARITHMETIC OPERATORS**

Arithmetic operators are used to perform arithmetic operations that include addition, subtraction, multiplication, division and also to find the remainder obtained when an integer is divided by another integer.

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder (Modulus) Operator

6) What are relational operators? Describe with an example.

Ans:

Relational Operator
• Relational operators are used to compare two values of the same types.
• After evaluation of a relational expression, the result produced is either TRUE or FALSE
• Relational operator include ==, !=, <, >, <= and >=.

Examples:

Relation Expression	Explanation	Result
5 = =5	5 is equal to 5 ?	TRUE
5 ! = 7	5 is not equal to 7?	TRUE
5 > = 5	5 is greater than or equal to 5?	TRUE
5 < = 4	5 is less than or equal to 4?	FALSE

7) What are logical operators? Describe with an example.

Ans:

LOGICAL OPERATORS

Logical operators are used for building compound conditions. We have seen before that a single condition is built using a relational operator in an expression. If we need to build more than one condition for some action to take place in programming, then we have to form compound condition.

Types of Logical Operators:

There are three types of logical operators. These are:

Operator	Definition
& &	AND
||	OR
!	NOT

AND operator (&&):

AND operator && takes two Boolean expressions as operands and produces the result TRUE if both of its operands are TRUE. It returns FALSE if any of the operands is FALSE. Table shows the truth table for AND operator.

Expression	Result
FALSE && FALSE	FALSE
FALSE && TRUE	FALSE
TRUE && FALSE	FALSE
TRUE && TRUE	TRUE

OR Operator (||):

OR operator accepts Boolean expression and returns true if at least one of the operands it true. Table shows that truth table for OR operator.

Expression	Result
FALSE || FALSE	FALSE
FALSE || TRUE	TRUE
TRUE || FALSE	TRUE
TRUE || TRUE	TRUE

NOT Operator (!):

NOT operator negates or reverses the value of Boolean expression. It makes it TRUE, if it is FALSE and FALSE if it is TRUE. Table presents the truth table for not operator.

Expression	Result
!(TRUE)	FALSE
!(FALSE)	TRUE

Examples of Logical Operators:

Logical Expression	Explanation	Result
3 < 4 && 7 > 8	3 is less than 4 AND is greater than 8?	FALSE
3 = = 4 || 3 > 1	3 is equal to 4 OR 3 is greater than 1?	TRUE
! (4 > 2 || 2 = = 2)	NOT (4 is greater than 2 OR 2 is equal to 2)?	FALSE

Example:

The expression: **! (a<b) w**ill be true if **a** is not less than **b**. In other words, the condition will be TRUE if **a** is greater than or equal to **b**. The same condition can also be written as **(a>=b)** which is easy to understand.

8) What is the difference between unary operators and binary operators?

Ans:

Unary Operators	Binary Operators
Unary operators are applied over one operand only	Binary operators require two operands to perform the operation
Example Logical not (!) operator has only one operand. Sign operator (-) is another example of a unary operator e.g. -5.	**Example** All the arithmetic operators, and relational operators are binary operators. The logical operator && and || are also binary operators.

9) What are the differences between '=' operators and '==' operators?

Assignment Operator (=)	Equal to Operators (= =)
Assignment operator is used to assign a value to a variable, or assign a value of variable to another variable.	In C language, = = operator is used to check for equality of two expressions
Equal sign (=) is used as assignment operator in C.	Double Equal sign (= =) is used as equal to operator
Consider the following example: int sum = 5;	Consider the following example: 3+2 = = 5;

10) What is meant by precedence of operators? Which operator has the highest precedence in C language?

Ans: If there are multiple operators in an expression, the question arises that with operator is evaluated first. To solve this issue, a precedence has been given to each operator (Table given below). An operator with higher precedence is evaluated before the operator with lower precedence. In case of equal precedence, the operator at left side is evaluated before the operator at right side.

Example:

Result = 18 / 2 * 3 + 7% 3 + (5 * 4);

Operator	Precedence
()	1
!	2
* , / , %	3
+ , -	4
> , < , > = , < =	5
= = , ! =	6
&&	7
||	8
=	9

Q5. Write down output of the following code segments.

	Code	Output

a.	# include<stdio.h>
	void main ()
	{
	int x = 2, y = 3, z = 6;
	int ans1, ans2, ans3;
	ans1= x / z * y;
	ans2 = y + z / y * 2;
	ans3 = z / x +x * y;
	printf("%d %d %d", ans1, ans2, ans3);
	}
		0 7 9

b.	# include<stdio.h>
	void main ()
	{
	printf (“nn \n\n nnn \nn\nt\t”);
	printf (“nn /n/n nn/n/n”);
	}
		**nn
		nnn
		n t
		nn/n/n nn/n**

c.	# include<stdio.h>
	void main ()
	{
	int a = 4, b;
	float c = 2.3;
	b=c*a;
	printf(“%d”,b);
	}
		9

d.	# include<stdio.h>
	void main ()
	{
	int a = 4 * 3 / (5+1) + 7 % 4;
	printf(“%d”,a);
	}
		5

e.	# include<stdio.h>
	
	void main ()
	{
	printf(“%d”,(((5 > 3) && (4 > 6)) || (7 > 3)));
	}
		1

Q6. Identify errors in the following code segments.

	Code	Identified Error/s	Correct Syntax

a.	# include<stdio.h>
	void main ()
	{
	int a , b = 13;
	b = a% 2;
	printf(“Value of b is : %d , b);
	}
		Trace error in line #3 (body of function) “closing” inverted commas missing	printf(“Value of b is : %d” , b);

b.	# include<stdio.h>
	void main ()
	{
	int a , b , c;
	printf(“Enter First Number: ”);
	scanf(“%d” , &a);
	printf(“Enter Second Number: ”);
	scanf(“%d”, &b);
	a + b = c;
	]
		Trace error in closing brace of body of function	Use curly braces **}** instead of that **]**

c.	# include<stdio.h>
	main ()
	{
	int num;
	printf(Enter number: ”
	);
	scanf(“%d, & num);
	};
		Trace error at the end of body of function (curly braces) statement terminator used};	Statement terminator ; can’t be used at the end of body of function

d.	#include<stdio.h>
	int main ()
	{
	float f;
	printf [“Enter First Number: ”];
	scanf(“%c”, &f);
	}
		Trace error in scanf (“%c”, &f);	Float data type always written with the format specifier %f

PROGRAMMING EXERCISES (A.B)

Exercise 1

The criteria for calculation of wages in a company is given below.

Basic Salary	**= Pay Rate Per Hour**	**x**	**Working Hours Of Employee**
Overtime Salary	**= Overtime Pay Rate**	**x**	**Overtime Hours Of Employee**
Total Salary	**= Basic Salary**	**+**	**Overtime Salary**

Write a program that should take working hours and overtime hours of employee as input. The program should calculate and display the total salary of employee.

Solution:

```
# include <stdio.h>
        void main ( )
{
        float working_hours,overtime_hours;
        float pay_per_hour,overtime_per_hour;
        float basic_salary,overtime_salary,Total_salary;
        printf("Enter working hours of employee\n");
        scanf("%f",&working_hours);
        printf("Enter pay rate per hour\n");
        scanf("%f",&pay_per_hour);
        printf("Enter overtime hours of employee");
        scanf("%f",&overtime_hours);
        printf("Enter overtime rate per hour of employee");
        scanf("%f",&overtime_per_hour);
        basic_salary=pay_per_hour*working_hours;
        overtime_salary= overtime_per_hour*overtime_hours;
        Total_salary=basic_salary+overtime_salary;
        printf("Total Salary=%f",Total_salary);
}
```

Exercise 2

Write a program that takes Celsius temperature as input, converts the temperature into Fahrenheit and shows the output. Formula for conversion of temperature from Celsius to Fahrenheit is: $F = \frac{9}{5}C + 32$

Solution:

```
# include <stdio.h>
        void main ( )
{
        float C,F;
        printf(" enter tempratue in celsius\n");
        scanf("%f",&C);
        F = 9/5*C+32;
        printf("Temprature in Farenheit=%f",F);
}
```

Exercise 3

Write a program that displays the following output using single *printf* statement:

*** * * ***

1 2 3 4

Solution:

```
# include <stdio.h>
void main ( )
{
printf("*\t* t*\t*\n1\t2\t3 t4");
}
```

Exercise 4

Write a program that displays the following output using single printf statement:

I am a Boy

I live in Pakistan

I am a proud Pakistani

Solution:

```
# include <stdio.h>
void main ( )
{
printf ("I am a boy\nI live in Pakistan\nI am a proud Pakistani");
}
```

Exercise 5

A clothing brand offer 15% discount on each item. A lady buys 5 shirts from this brand. Write a program that calculate total price after discount and amount of discount availed by the lady. Original prices of the shirts are:

Shirt1 = 423

Shirt2 = 320

Shirt3 = 270

Shirt4 = 680

Shirt5 = 520

Note: Use 5 variables to store the prices of shirts.

Solution:

```
# include <stdio.h>
void  main( )
{
int shirt1=423, shirt2=320, shirt3=270, shirt4=680, shirt5=520;
float Total_Price, Total_Discount, Final_Price;
printf("First Shirt price= %d \t\tDiscount is %.2f \tPayable price= %.2f\n",shirt1, (shirt1*0.15),
(shirt1-(shirt1*0.15)));
printf("Second Shirt price= %d \tDiscount is %.2f \tPayable price= %.2f\n",shirt2,
(shirt2*0.15), (shirt2-(shirt2*0.15)));
printf("Third Shirt price= %d \t\tDiscount is %.2f \tPayable price= %.2f\n",shirt3,
```

```
(shirt3*0.15), (shirt3-(shirt3*0.15)));
printf("Fourth Shirt price= %d \tDiscount is %.2f \tPayable price= %.2f\n",shirt4,
(shirt4*0.15), (shirt4-(shirt4*0.15)));
printf("Fifth Shirt price= %d \t\tDiscount is %.2f \tPayable price= %.2f\n",shirt5,
(shirt5*0.15), (shirt5-(shirt5*0.15)));
printf("\n\n---------------------------------------------------------------------------------
---------\n");
Total_Price= shirt1 + shirt2 + shirt3 + shirt4 + shirt5;
Total_Discount=(shirt1*0.15)+(shirt2*0.15)+(shirt3*0.15)+(shirt4*0.15)+(shirt5*0.15);
Final_Price = (shirt1-(shirt1*0.15))+(shirt2-(shirt2*0.15))+(shirt3-(shirt3*0.15))+(shirt4-
(shirt4*0.15))+(shirt5-(shirt5*0.15));
printf("Total price: %.2f \tTotal Discount is %.2f \tFinal price: %.2f\n",Total_Price,
Total_Discount, Final_Price);
}
```

Exercise 6

Write a program that swaps the values of two integer variables without help of any third variable.

Solution:

```
# include <stdio.h>
void main( )
{
int a, b;
printf("Enter a\n");
scanf("%d", &a);
printf("Enter b\n");
scanf("%d", &b);
a = a - b;
b = a + b;
a = b - a;
printf("After swapping, a = %.2d\n", a);
printf("After swapping, b = %.2d", b);
}
```

Exercise 7

Write a program that takes a 5-digit number as input, calculates and displays the sum of first and last digit of number.

Solution:

```
# include <stdio.h>
void main( )
{
int number;
printf("Enter a number with length 5 digits=\n");
scanf("%d",& number);
printf("The sum of first and 5th digit is=%d", (number/10000)+(number%10));
}
```

Exercise 8

Write a program that takes monthly income and monthly expenses of the user like electricity bill, gas bill, food expense. Program should calculate the following:

- **Total monthly expenses**
- **Total yearly expenses**
- **Monthly savings**
- **Yearly saving**
- **Average saving per month**
- **Average expense per month**

Solution:

```
#include <stdio.h>
void main()
{
   float monthly_income, electricity_bill, gas_bill, food_expense, monthly_Expenses,
monthly_savings;
   printf("Please enter your Monthly Income=");
   scanf("%f",&monthly_income);
   printf("Please enter you Monthly Expenses=");
   printf("\n\n\tYour Electricity Bill=");
   scanf("%f",&electricity_bill);
   printf("\tYour Gas Bill=");
   scanf("%f",&gas_bill);
   printf("\tYour Food Expense=");
   scanf("%f",&food_expense);
   monthly_Expenses=electricity_bill+gas_bill+food_expense;
   printf("\n\n\tTotal Monthly Expenses are=%f", monthly_Expenses);
   printf("\n\tTotal Yearly Expenses are=%f", (monthly_Expenses)*12);
   monthly_savings =monthly_income-monthly_Expenses;
   printf("\n\n\tYour Monthly savings is=%f", monthly_savings);
   printf("\n\tYour Monthly savings is=%f", monthly_savings*12);
   printf("\n\n\tAverage saving per month=%f", monthly_savings);
   printf("\n\tAverage expense per month=%f", monthly_Expenses);
}
```

Exercise 9

Write a program that takes a character and number of steps as input from user. Program should then jump number of steps from the character.

Sample output:

Enter character: a

Enter steps: 2

New character: c

Solution:

```
# include <stdio.h>
void main( )
{
char c;
int steps,x;
printf("Enter character=");
scanf("%c",&c);
printf("Enter Steps=");
scanf("%d",&steps);
 x = c + steps;
printf("New Character: %c",x);
}
```

Exercise 10

Write a program that takes radius of a circle as input. The program should calculate and display the area of circle.

Solution:

```
# include <stdio.h>
void  main( )
{
float r, Area;
printf(“Enter the radius of circle”);
scanf(“%f”,&r);
Area = 3.14*r*r;
printf(“The area of the circle is %f\n”,Area);
}
```

ANSWER KEY

2.1 INPUT / OUTPUT (I/O) FUNCTION

2.1.1 PRINTF ()

2.1.2 FORMAT SPECIFIERS

2.1.3 SCANF ()

2.1.4 GETCH ()

2.1.5 STATEMENT TERMINATOR

2.1.6 ESCAPE SEQUENCE

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	D	D	A	A	C	B	B	D	C	A	C	D	A	B
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	C	C	B	C	B	C	A	D	A	B	A	A	C	A
31	32	33	34	35	36	37	38							
B	B	C	B	B	C	B	A							

2.2 OPERATORS OF C LANGUAGE

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
D	C	A	A	A	C	A	B	C	C	C	B	C	B	A
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	C	D	A	A	A	A	C	D	A	D	A	C	B	B
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
D	C	A	B	B	B	C	B	B	B	C	D	A	B	A
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
B	A	B	B	C	A	B	A	A	B	B	C	C	B	B
61	62	63												
C	B	D												

CH # 3 CONDITIONAL LOGIC

3.1 CONTROL STATEMENT

SHORT QUESTIONS

Q.1 Define Control Statement. (K.B)

Ans:

CONTROL STATEMENT

"Control Statement controls the flow of execution of a program" Sometimes we need to execute one set of instructions if a particular condition is **TRUE** and another set of instructions if the condition is **FALSE**. Moreover, sometimes we need to repeat a set of statements for a number of times. We can control the flow of program execution through control statements. There are three types of control statements in C language.

1) Sequential Control Statements
2) Selection Control Statements
3) Repetition Control Statements

Q.2 Name Types of Control Statement. (K.B)

Ans:

TYPES OF CONTROL STATEMENT

There are three types of control statements in C language.

1. Sequential Control Statements
2. Selection Control Statements
3. Repetition Control Statements

Sequential Control:

Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.

Selection Statements:

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

Two types of selection statements are:

1. If statement
2. If-else statement

Repetition Control:

The control structure which keeps on repeating a statement or a set of statements upto a fixed number of time or until an associated condition remains true.

Two types of repetition statements are:

1. FOR statement
2. WHILE statement
3. DO-WHILE statement

MUTLIPLE CHOICE QUESTIONS

1. _____ is the default control structure: (K.B)
(A) Sequence (B) Selection (C) Repetition (D) None of these

2. In _____ control structure the instructions are executed according to ascending order. (K.B)
(A) Sequence (B) Selection (C) Repetition (D) None of these

3. _____ statement are executed on basis of condition. (K.B+U.B)
(A) Sequential (B) Selection (C) both (D) None of these

4 Condition is always written in _____. (K.B)
(A) Quotes " " (B) Parentheses () (C) Braces{} (D) None of these

5. A condition can be _____ expression. (K.B)
(A) Relational (B) Logical (C) Arithmetic (D) All of these

3.2 SELECTION STATEMENTS

3.2.1 IF STRUCTURE
3.2.2IF-ELSE STRUCTURE
3.2.3NESTED SELECTION STRUCTURES.

LONG QUESTIONS

1. Define Selection Statement. Explain IF Statement in detail with an example. (K.B+U.B)

Ans:

SELECTION STATEMENTS

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

Two types of selection statements are:

1. if statement
2. if-else statement

If statement:

C language provides **if statement** in which we specify a condition, and associate a code to it. The code gets executed if the specified condition turns out to be true, otherwise the code does not get executed.

Structure of if statement:

if statement has the following structure in C language:

if (condition)
Associated Code

1. In the given structure, if is a keyword that is followed by a condition inside parentheses ( ).
2. A condition could be any valid expression including arithmetic expressions, relational expressions, logical expressions, or a combination of these.

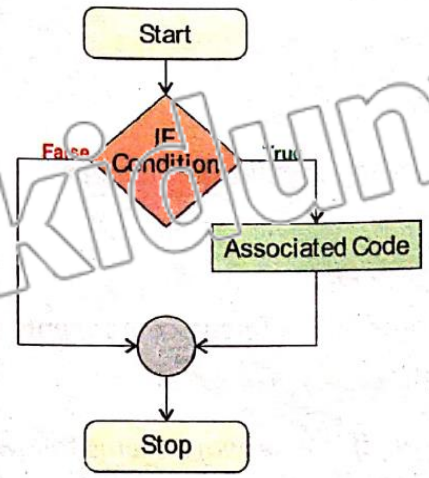
Example:

Here are a few examples of valid expressions that can be used as condition.

a- 5 (TRUE)
b- 5 + 4 (TRUE)
c- 5 – 5 (FALSE)

Any expression that has a non-zero value calculates to true, e.g. expressions a and b above produce a true value, but the expression c produces a false value. The expression can also include variables, in that case values inside the variables are used to calculate the true/false value of the expression.

3. The associated code is any valid C language set of statements. It may contain one or more statements. The following flow chart shows the basic flow of an if statement.



If we want to associate more than one statements to an **if statement**, then they need to be enclosed inside a **{ }** block, but if we want to associate only one statement, then although it may be enclosed inside **{ }** block, but it is not mandatory.

Example:

```
#include < stdio.h>
void main ( )
{
int a = 12;
if (a % 2 == 0)
{
printf (“The variable a contains an even value.”);
printf (“\nYou are doing a great job.”);
}
}
```

Output:

The variable a contains an even value.
You are doing a great job.

2. Define Selection Statement. Explain IF-ELSE Statement in detail with an example.(K.B+U.B)

Ans:

SELECTION STATEMENTS

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

Two types of selection statements are:

1) if statement
2) if-else statement

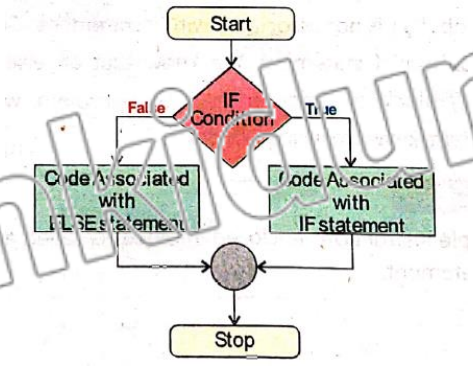
IF-ELSE STATEMENT

“**if-else** statement executes the set of statements under **if** statement if a condition is true and executes the set of statements under else otherwise”

General structure of the if-else statement is as follows:

```
if (condition)
Associated Code
else
Associated Code
```

Associated code of **if** statement is executed if the condition is TRUE, otherwise the code associated with **else** statement is executed. Following flow chart shows the structure of **if-else** statement.



An **if** statement may not have an associated **else** statement, but an **else** statement must have an **if** statement to which it is associated. Before **else** keyword, if there are multiple statements under if, then they must be enclosed inside the **{ }** block, otherwise compiler issues an error.

Example:

```
#include < stdio.h>
void main ( )
{
int a = 15;
if (a % 2 == 0)
{
printf ("The variable a contains an even value.");
printf ("\n You are doing a great job.");
}
else
printf ("The variable a contains an odd value.");
}
```

Output:

The variable a contains an odd value.

3. Define nested selection structure. Explain it with suitable examples. (K.B+U.B)

Ans Conditional statements within conditional statements are called nested selection structure. All the following structures are valid nested selection structures.

if (condition1 is true)
  if (condition2 is true)
  Associated code
  else
  Associated code
	if (condition1 is true)
	  if (condition2 is true)
	  Associated code
	  else
	  if (condition3 is true)
	  Associated code

if (condition1 is true)
  if (condition2 is true)
  Associated code
  else
  Associated code
  else
  if (condition3 is true)
  Associated code
	if (condition1 is true)
	  if (condition2 is true)
	  Associated code
	  else
	  Associated code
	  else
	  if (condition3 is true)
	  Associated code
	  else
	  Asscoted code

Example:

An electricity billing company calculates the electricity bill according to the following formula

Bill Amount = Number of Units Consumed **X** Unit Price

There are two types of electricity users i.e. Commercial and Home Users. For home users the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 12
Units > 200 but Units < = 400	Rs 15
Units > 400	Rs 20

For commercial users, the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 15
Units > 200 but Units < = 400	Rs 20
Units > 400	Rs 24

Write a program that takes the type of consumer and number of units consumed as input. The program then displays the electricity bill of the user.

Program:

```
#include<stdio.h>
void main()
{
        int units, unit_price, bill;
        char user_type;
        printf("Please enter h for home user and c for commercial user:  ");
        scanf("%c", &user_type);
        printf("Please enter the number of  units consumed:  ");
        scanf("%d", &units);
        if(units < = 200)
                if(user_type == 'h')
                        unit_price = 12;
                else if(user_type == 'c')
                        unit_price = 15;
        else if(units > 200 && units <= 400)
                if(user_type == 'h')
                        unit_price = 15;
                else if(user_type  ==  'c')
                        unit_price = 20;
        else
                if(user_type == 'h')
                        unit_price = 15;
                else if(user_type == 'c')
                        unit_price = 24;
        bill = units * unit_price;
        printf("Your electricity bill is %d",  bill);
}
```

The code associated with an ***if statement*** or with an ***else*** statement can be any valid 'C' language set of statements. It means that inside an ***if*** block or inside an ***else*** block, we can have other ***if*** statements or ***if-else*** statements. It also means that inside those inner ***if*** statement or ***if-else*** statements we can have even more ***if*** statements or ***if-else*** statements and so on.

SHORT QUESTIONS

Q.1 What are selection statements? (K.B)

Ans:

SELECTION STATEMENTS

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements. Two types of selection statements are:

1. if statement 2. if-else statement

Q.2 Write the name of types of selection statements. (K.B)

Ans:

TYPES SELECTION STATEMENTS

Two types of selection statements are:

1. if statement 2.if-else statement

Q.3 Define Condition. (K.B)

Ans: A condition could be any valid expression including the arithmetic expressions, relational expressions, logical expression, or a combination of these. Condition always evaluates in TRUE or FALSE.

Q.4 Define IF statement. (K.B)

Ans:

IF STATEMENT

"if-else statement executes the set of statements under if statement if a condition is TRUE and executes the set of statements under else otherwise"

Syntax:

if (condition)
Associated Code

Q.5 Give Syntax of IF Statement. (K.B+U.B+A.B)

Ans:

SYNTAX OF IF STATEMENT

if statement has the following structure in C language:

if (condition)
Associated Code

Q.6 What do you know about associated code? (K.B+U.B)

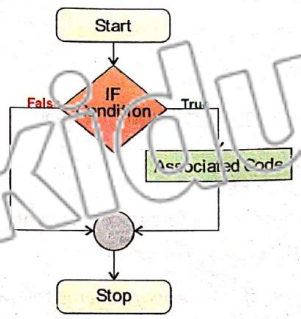
Ans:

ASSOCIATED CODE

The associated code is any valid C language set of statements. It may contain one or more statements.

Q.7 Make a Flow chart to explain IF structure. (K.B+U.B)

Ans:



Q.8 Define IF-ELSE statement. (K.B)

Ans:

IF-ELSE STATEMENT

"**if-else** statement executes the set of statements under **if** statement if a condition is TRUE and executes the set of statements under else otherwise"

Q.9 Give syntax of IF-ESLE Statement (K.B+U.B)

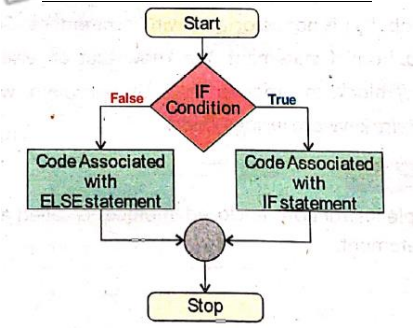
Ans: **IF-ESLE STATEMENT**

General structure of the if-else statement is as follows:

```
if (condition)
Associated Code
else
Associated Code
```

Q.10 Make a flow chart to explain if-else structure. (K.B+U.B)

Ans: **IF-ELSE STRUCTURE**



Q.11 Define block. (K.B)

Ans: A set of multiple instructions enclosed in braces is called a block or compound statement.

Q.12 What will happen if the multiple statements of IF statement are not enclosed in { } before use of ELSE statement. Show with an example.

Ans: Before else keyword, if there are multiple statements under if, then they must be enclosed inside the { } block, otherwise compiler issues an error. In order to understand this concept, let's look at the following example.

Example:

```
#include < stdio.h>
void main ( )
{
int a = 15;
if (a % 2 == 0)
{
printf ("The variable a contains and even value.");
printf ("/nYou are doing a great job.");
}
else
printf ("The variable a contains an odd value.");
}
```

Q.13 Define IF-ELSE-IF statement. (K.B)

Ans: **IF-ELSE-IF STATEMENT**

if-else-if statement also known as **else-if** statement. It is also a type of selection statement. It is used in a program when we have to use multiple conditions with an if statement.

Syntax:

```
if (condition 1)
        Code to execute if condition 1 is true;
else if (condition 2)
Code to execute if condition 1 is false but condition 2 is true;
```

.
.
.
.
.
else if (condition N)
code to execute if all previous conditions are false but condition N is true;
else
Code to execute if all the conditions are false;

Q.14 Give Syntax of IF-ELSE-IF Statement. (K.B+U.B)

Ans: **SYNTAX OF IF-ELSE-IF STATEMENT**

General syntax of the if-else-if statement is as follows:

```
if (condition 1)
Code to execute if condition 1 is TRUE;
else if (condition 2)
Code to execute if condition 1 is FALSE but condition 2 is TRUE;
.
.
.
.
.
else if (condition N)
code to execute if all previous conditions are FALSE but condition N is TRUE;
else
Code to execute if all the conditions are FALSE;
```

Q.15 Define Nested Selection Structure. (K.B)

Ans: **SELECTION STRUCTURE**

Conditional statements within conditional statements are called nested selection structures.

Q.16 Write the syntax of nested selection structure. (K.B)

Ans: **SYNTAX OF NESTED SELECTION STRUCTURE**

All the following structures are valid nested selection structures.

if (condition1 is true)
  if (condition2 is true)
  Associated code
  else
  Associated code
	if (condition1 is true)
	  if (condition2 is true)
	  Associated code
	  else
	  if (condition3 is true)
	  Associated code

if (condition1 is true)
  if (condition2 is true)
  Associated code
  else
  Associated code
  else
  if (condition3 is true)
  Associated code
	if (condition1 is true)
	  if (condition2 is true)
	  Associated code
	  else
	  Associated code
	  else
	  if (condition3 is true)
	  Associated code
	  else
	  Asscoted code

MUTLIPLE CHOICE QUESTIONS

1. _____ is the default control structure: (K.B)

(A) Sequence (B) Selection (C) Repetition (D) None of these

2. In _____ control structure the instructions are executed according to ascending order. (K.B+U.B)

(A) Sequence (B) Selection (C) Repetition (D) None of these

3. _____ statement are excited on basis of condition. (K.B+U.B)

(A) Sequential (B) Selection (C) Both A & B (D) None of these

4. Condition is always written in _____. (K.B)

(A) Quotes " " (B) parentheses (C) Braces {} (D) None of these

5. A condition can be _____ expression. (K.B)

(A) Relational (B) Logical (C) Arithmetic (D) All of these

6. True is indicated by: (K.B)

(A) 1 (B) 0 (C) Both A & B (D) None of these

7. False is indicated by: (K.B)

(A) 0 (B) 1 (C) Both A & B (D) None of these

8. _____ is a valid C language set of statement/s. (K.B)

(A) Associated Code (B) Condition (C) Expression (D) None of these

9. If we want to more than one statement with if statement then they needs to be enclosed with _____. (K.B+U.B)

(A) {} (B) ( ) (C) " " (D) None of these

10. Using _____ can improve the readability of program. (K.B)

(A) Spaces (B) Tag (C) Enter (D) None of these

11. If we don't use {} before Else statement if we have multiple statements, then _____.

(A) Complier issue an error (B) Complier ignore it

(C) Complier ignores else (D) None of these

12. Set of multiple statement enclosed in braces is called _____. (K.B)

(A) Multiple (B) Compound Statement (C) Group (D) None of these

13. A _____ is a group of statement enclosed in {} called _____ (K.B)

(A) Block (B) Group (C) Multiple (D) All of these

14. Code to execute after else statement in If -else- if statement only when. (K.B+U.B)

(A) 1st condition is true (B) 2nd condition is true

(C) 3rd condition is true (D) 4th condition is true

15. It is common mistake to omit_____. (K.B+U.B+A.B)

(A) keywords (B) Braces (C) Parenthesis (D) None of these

16. _____ is applicable in only limited scenarios. (K.B)

(A) if-else (B) if (C) if-else-if (D) switch –case

PROGRAMMING TIME (A.B)

Programming Time 3.1

Problem:
Write a program in C language that takes the percentage of student as an input and displays "PASS" if the percentage is above 50.

```
#include  <stdio.h>
void main()
{
        float percentage;
        printf ("Enter the percentage: ");
        scanf  ("%f, &percentage);
        if (percentage  >  50)
                printf ("PASS\n");
}
```

Output:
On the input 47, program simply ends because 47 is less than 50 and the condition turns false.

```
Enter the percentage: 47
```

When 67.3 is entered as an input, "PASS" gets printed on console because condition is true, as 67.3 is greater than 50.

```
Enter the percentage: 67.3
PASS
```

Programming Time 3.2

Problem:
A marketing firm calculates the salary of its employees according to the following formula.

Gross Salary = Basic Salary + (Number of Items Sold X8) + Bonus

If the number of sold items are more than 100 and the number of broken items are 0, then bonus is Rs. 10000, otherwise bonus is 0.
Write a program that takes basic salary, the number of sold and broken items as input from user, then calculates and displays the gross salary of the employee.

Program:

```
#include<stdio.h>
void  main()
{
```

```
        int basic_salary, items_sold, items_broken, gross_salary;
        int bonus = 0;
        printf("Enter the basic salary: ");
        scanf("%d, &basic_salary);
        printf("Enter the number of items sold: ");
        scanf("%d", &items_sold);
        printf("Enter the number of items broken: ");
        scanf("%d, &items_broken);
        if (items_sold > 100 && items_broken == 0)
                bonus = 10000;
        gross_salary = basic_salary + (items_sold * 8) + bonus;
        printf("Gross salary of the employee is %d", gross_salary);
}
```

Description:

In the above example, bonus is initialized to 0 because if the number of sold items are not more than 100, then automatically bonus is considered 0. Inside the if statement, it is checked that whether the number of sold items are greater than 100. If so, the bonus is assigned 10000. It is to be noted that gross salary is calculated outside the if block, because whether the number of sold items are more than 100 or not, the gross salary must be calculated.

Programming Time 3.3

Problem:

Write a program that takes percentage marks of student as input and displays his grade. Following table shows grades distribution criteria.

Percentage	Grade
80% and above	A
70% – 80%	B
60% – 70%	C
50% – 60%	D
Below 50%	F

Program:

```
#include<stdio.h>
void main()
{
        float percentage;
        printf ("Enter the percentage:   ");
        scanf ("%f", &percentage);
        if (percentage >= 80)
                printf   ("A\n");
        else if (percentage   >= 70)
                printf   ("B\n");
        else if (percentage   >= 60)
                printf   ("C\n");
        else if (percentage   >= 50)
                printf   ("D\n");
        else
                printf   ("F\n");
}
```

Programming Time 3.4

Problem:

An electricity billing company calculates the electricity bill according to the following formula.

Bill Amount = Number of Units Consumed X Unit Price

There are two types of electricity users i.e. Commercial and Home Users. For home users the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 12
Units > 200 but Units < = 400	Rs 15
Units > 400	Rs 20

For commercial users, the unit price varies according to the following:

Units Consumed	Unit Price
Units < = 200	Rs 15
Units > 200 but Units < = 400	Rs 20
Units > 400	Rs 24

Write a program that takes the type of consumer and number of units consumed as input. The program then displays the electricity bill of the user.

Program:

```
#include<stdio.h>
void main()
{
        int units, unit_price, bill;
        char user_type;
        printf("Please enter h for home user and c for commercial user:  ");
        scanf("%c", &user_type);
        printf("Please enter the number of  units consumed:  ");
        scanf("%d", &units);
        if(units < = 200)
                if(user_type == 'h')
                        unit_price = 12;
                else if(user_type == 'c')
                        unit_price = 15;
        else if(units > 200 && units <= 400)
                if(user_type == 'h')
                        unit_price = 15;
                else if(user_type  ==  'c')
                        unit_price = 20;
        else
                if(user_type == 'h')
                        unit_price = 15;
                else if(user_type  ==  'c')
                        unit_price = 24;
        bill = units * unit_price;

        printf("Your electricity bill is %d",  bill);
}
```

Programming Time 3.5

Program:

Write a program that displays larger one out of the three given number.

Program:

```
include <stdio.h>
void main()
{
        int n1, n2, n3;
        printf (“Enter three numbers”);
        scanf (“%d%d%d”, &n1, &n2, &n3);
        if (n1 > n2 &&n1 > n3)
                printf (“The largest number is %d”, n1);
        else if (n2  > n3 && n2  > n1)
                printf (“The largest  number is %d”, n2);
        else
        printf (“The largest  number is %d”, n3);
}
```

Programming Time 3.6

Problem:

Write a program that calculates the volume of cube, cylinder or sphere, according to the choice of user.

Program:

```
#include<stdio.h>
void main ()
{
        int choice;
        float volume ;
        printf (“Find Volume\n”);
        printf (“1.Cube\n2.Cylinder\n3.Sphere\nEnter your choice: ");
        scanf  (“%d”, &choice);
        if (choice == 1)
```

```
{
        float length;
        printf (“Enter Length: ”);
        scanf (“%f”, &length);
        volume = length * length * length;
        printf (“Volume is %f”, volume);
}
else if (choice == 2)
{
        float length1, radiusl;
        printf (“Enter Length: ”);
        scanf (“%f”, &length1);
        printf (“Enter Radius: ”);
        scanf (“%f”, &radius1);
        volume = 3.142 * radius1 * radius1 * length1;
        printf (“Volume is %f”, volume);
}
        else  if (choice == 3)
        {
                float radius;
                printf (“Enter Radius: ”);
                scanf (“%f”, &radius);
                volume = 3.142 * radius * radius * radius;
                printf (“Volume is %f”, volume);
        }
        else
                printf (“Invalid Choice”);
}
```

SOLVED ACTIVITIES (A.B)

ACTIVITY 3.1

Write a program that takes the age of a person as an input and displays "Teenager" if the age lies between 13 and 19.

Solution:

```
# include <stdio.h>
void main ()
{
int age;
printf("Enter your age");
scanf("%d",&age);
if(age>=13&&age<=19)
printf("Teenager");
else
printf("Not Teenager");
}
```

ACTIVITY 3.2

Write a program that takes year as input and displays "Leap Year" if the input year is leap year. Leap years are divisible by 4.

Solution

```
# include <stdio.h>
void main ()
{
int year;
printf("Enter a year");
scanf("%d",&year);
if(year%4==0)
printf("Leap Year");
else
printf("Not Leap Year");
}
```

ACTIVITY 3.3

Write a program that takes the value of body temperature of a person as an input and displays "You have fever." if body temperature is more than 98.6 otherwise displays "You don't have fever."

Solution

```
# include <stdio.h>
void main ( )
{
float  body_temp;
printf("Enter your body temperature");
scanf("%f",&body_temp);
if(body_temp>98.6)
printf("You have Fever");
else
printf("you don't have fever");
}
```

ACTIVITY 3.4

The eligibility criteria of university for its different undergraduate student programs is as follows:
BSSE Program : 80% or more marks in Intermediate
BSCS Program : 70% or more marks in Intermediate
BSIT Program : 60% or more marks in Intermediate
Otherwise the university do not enroll a student in any of its programs.
Write a program that takes the percentage of Intermediate marks and tells for which programs the student is eligible to apply.

Solution

```
# include <stdio.h>
void main ( )
{
float marks_percentage;
printf("Enter the percentage of intermediate marks");
scanf("%f",&marks_percentage);
if(marks_percentage>=80)
printf("You are eligible for BSSE program");
else if(marks_percentage>=70&&marks_percentage<80)
printf("you are eligible for BSCS program");
else if(marks_percentage>=60&&marks_percentage<70)
printf("You are eligible for BSIT");
else
printf("You are not Eligible for any program");
}
```

ACTIVITY 3.5

Write a program that takes two integers as input and asks the user to enter a choice from 1 to 4. The program should perform the according to the given table

CHOICE	OPERATION
1	Addition
2	Subtraction
3	Multiplication
4	Division

Solution

```
# include <stdio.h>
void main ( )
{
int num_1,num_2;
int choice;
printf("Enter first number");
scanf("%d",&num_1);
printf("Enter second number");
scanf("%d",&num_2);
printf("Enter your choice press 1 for Addition press 2 for Subtraction
press 3 for Multiplication press 4 for Division ");
if(choice==1)
printf("Addition=%d",num_1+num_2);
else if(choice==2)
printf("Subtraction=%d",num_1-num_2);
else if(choice==3)
printf("Multiplication=%d",num_1*num_2);
else if(choice==4)
printf("Division=%d",num_1/num_2);
else
printf("Invalid Input");
}
```

ACTIVITY 3.6

Write a program that finds and display s area of a triangle, parallelogram, rhombus or trapezium according to the choice of user.

Solution

```
# include <stdio.h>
void main ( )
{
int choice;
printf("Enter your choice to find the area of figure\n 1 for Triangle\n 2 for Parallelogram\n 3
                for Rhoumbus\n 4 for Trapezium");
scanf("%d", &choice);
if(choice==1)
{ //This block will calculate the area of triangle
float b,h,area;
printf("Enter the base & height of Triangle");
scanf("%f%f", &b,&h);
area=1/2*b*h;
printf("The area of figure=%f",area);
}
else if(choice==2)
{ //This block will calculate the area of Parallelogram
float b,h,area;
printf("Enter the base & height of Parallelogram");
scanf("%f%f", &b,&h);
area=b*h;
printf("The area of figure=%f",area);
}
else if(choice==3)
{ //This block will calculate the area of Rhoumbus
float b,h,area;
printf("Enter the base & height of Rhoumbus");
scanf("%f%f", &b,&h);
area=b*h;
printf("The area of figure=%f",area);
}
else if(choice==4)
{ //This block will calculate the area of Trapezium
float h,side1,side2,area;
printf("Enter the height,side1 & side 2 of Trapezium");
scanf("%f%f%f",&h,&side1, &side2);
area=1/2*(side1+side2)*h;
printf("The area of figure=%f",area);
}
else
printf("You have enter the wrong choice");
}
```

EXERCISE

Q1. Multiple Choice Questions.

1) Conditional logic helps in _____. (K.B)
(A) Decisions (B) Interactions (C) Traversing (D) All of these

2) _____ statements describe the sequence in which statements of the program should be executed. **(K.B)**
(A) Loop (B) Conditional (C) Control (D) All of these

3) In if statement, what happens of condition is false? (K.B)
(A) Program crashes (B) Index Out of Bound Error
(C) Further code executes (D) Compiler asks to change condition

4) (K.B)
```
int a = 5 ;
if (a < 10)
a++ ;
else
if (a > 4)
a --;
```
Which one of the following statements will execute?
(A) a++ (B) b--; (C) both (A) and (B) (D) None of these

5) Which of the following is the condition to check 'a' is a factor of 'c'? (K.B+U.B)
(A) a % c = = 0 (B) c % a = = 0 (C) a*c = = 0 (D) a + c = =0

6) A condition can be any _____ expression. (K.B+U.B)
(A) arithmetic (B) relational
(C) logical (D) arithmetic, relational or logical

7) An if statement inside another if statement is called _____ structure. (K.B)
(A) nested (B) boxed (C) repeated (D) decomposed

8) A set of multiple instruction enclosed in braces is called a _____. (K.B+U.B)
(A) box (B) list (C) block (D) job

ANSWER KEY

1	2	3	4	5	6	7	8
A	D	C	A	B	D	A	C

Q2. Define the following terms. (K.B)

1) Control Statements

Ans:

CONTROL STATEMENTS

"Control Statement controls the flow of execution of a program"

Sometimes we need to execute one set of instructions if a particular condition is true and another set of instructions if the condition is false. Moreover, sometimes we need to repeat a set of statements for a number of times. We can control the flow of program execution through control statements. There are three types of control statements in C language.

1. Sequential Control Statements
2. Selection Control Statements
3. Repetition Control Statements

2) Selection Statements

Ans:

SELECTION STATEMENTS

The statements which help us to decide which statements should be executed next, on the basis of conditions, are called selection statements.

Two types of selection statements are:

1. if statement
2. if-else statement

3) Sequential Statements

Ans: **SEQUENTIAL STATEMENTS**

Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.

4) Condition

Ans: A condition could be any valid expression including the arithmetic expressions, relational expressions, logical expression, or a combination of these. Condition always evaluates in true or false.

5) Nested Selection Structure

Ans: **NESTED SELECTION STRUCTURE**

A selection statement within another selection statements is known as Nested Selection Structure. The general structure of an if-else statement given below:

if (condition)
Associated Code
else
Associated Code

Q3. Briefly Answer the Following. **(K.B+U.B)**

1) Why do we need selection statements?

Ans: **SELECTION STATEMENTS**

The selection statements help us to decide which statements should be executed next, on the basis of conditions. These statements allows us to choose between the alternative program statements.

2) Differentiate between sequential statements and selection statements.

Ans: **STATEMENTS AND SELECTION STATEMENTS**

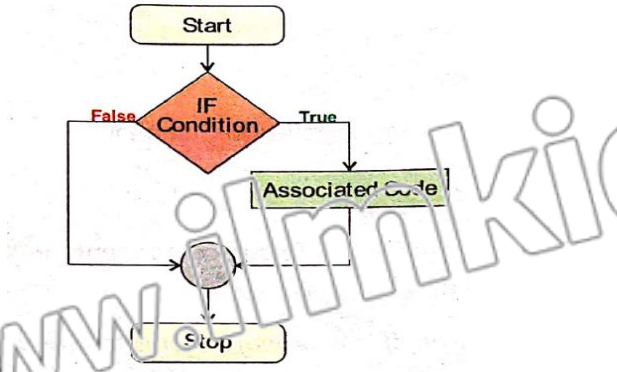
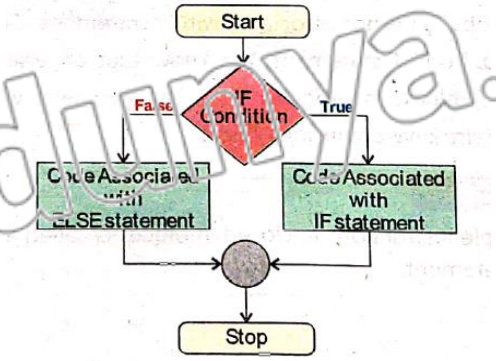
Sequential Statements	Selection Statements
Sequential control is the default control structure in C language. According to the sequential control, all the statements are executed in the given sequence.	The selection statements help us to decide which statements should be executed next, on the basis of conditions. These statements allows us to choose between the alternative program statements.

3) Differentiate between IF statements and IF-ELSE statement with an example.

Ans: **IF STATEMENTS AND IF ELSE STATEMENT**

IF STATEMENT	IF ELSE STATEMENT

if statement:
Definition
C language provides if statement in which we specify a condition, and associate a code to it. The code gets executed if the specified condition turns out to be TRUE, otherwise the code does not get executed.
Structure of if statement:
If statement has the following structure in C language:
If (condition)
Associated Code
In the given structure, if is a keyword that is followed by a condition inside parentheses ( ). A condition could be any valid expression including arithmetic expressions, relational expressions, logical expressions, or a combination of these.
	if-else Statement:
	Definition
	"if-else statement executes the set of statements under if statement if a condition is TRUE and executes the set of statements under else otherwise"
	Structure of if else statement:
	General structure of the if-else statement is as follows:
	if (condition)
	Associated Code
	else
	Associated Code
	Associated code of if statement is executed if the condition is TRUE, otherwise the code associated with else statement is executed. Following flow chart shows the structure of if-else statement.

Flowchart of if statement	Flowchart of if else statement
	

Example:
#include < stdio.h>
void main ( )
{
int a = 12;
if (a % 2 = = 0)
printf ("The variable a contains an even value.");
printf ("\nYou are doing a great job.");
}
}
	Example:
	#include < stdio.h>
	void main ( )
	{
	int a = 15;
	if (a % 2 == 0)
	{
	printf ("The variable a contains an even value.");
	printf ("/nYou are doing a great job.");
	}
	else
	printf ("The variable a contains an odd value.");
	}

4) What is the use of nested selection structures?

Ans: **SELECTION STRUCTURES**

Definition:

Conditional statements within conditional statements are called nested selection structures.

Format of Nested Selection Structure

if (condition 1 is TRUE)
if (condition 2 is TRUE)
associated code
else
associated code
else

5) Write the structure of if statement with brief description

Ans: **STRUCTURE OF IF STATEMENT**

If statement has the following structure in C language:

if (condition)
Associated Code

In the given structure, if is a keyword that is followed by a condition inside parentheses ( ). A condition could be any valid expression including arithmetic expressions, relational expressions, logical expressions, or a combination of these.

Q4. Identify the errors in following code segments. Assume that variables have already been declared. (K.B+U.B+A.B)

Ans:

Sr #	Program	Errors
(a)	if(x≥10) printf("Good");	Error expected in **if statement** we cannot write expression ≥ like that instead of that we have to write >=

(b)	if (a<b && b<C);
	sum = a+b+c;
	else
	multiply = a*b*c;
		Error expected in **if statement** we cannot use statement terminator; at the end of if statement.

(c)	if(a < 7 < (B) printf("7");	Error expected in **if statement** Logical operator (&& / ||) is not used in if statement.

(d)	if(a ==b &| x==y)
	flag = true;
	else
	flag = false;
		Error expected in **if statement** syntax of AND operator is not right logical operator cannot be write &| like that instead of that it can be write && like that.

e)	If(sum == 60 || product == 175)
	printf("Accepted %(C),sum);
	else
	if(sum>=45 || product > 100)
	printf("considered %d" + sum);
	else
	printf("Rejected");
		• Error expected in **line#2** (printf function) inverted comma's (string literals) are not used after format specifier.
		• Error expected in **line#5** (printf function) + sign is not allowed you have to use comma (,) at that place.

Q5. Write down output of the following code segments. (K.B+U.B+A.B)

Ans:

Sr #	Program	Output

(a)	int a = 7 , b = 10;
	a = a + b;
	if (a > 20 && b < 20)
	b = a + b ;
	printf ("a = %d , b = %d", a , b) ;
		a = 17
		b = 10

(b)	int x = 45 ;
	if (x + 20 * 7 = = 455)
	print f ("Look's Good");
	else
	print ("Hope for the Best");
		Look's Good

(c)	Char c1 = 'Y' , c2 = 'N' ;
	int n1 = 5, n2 = 9;
	n1 = n1 + 1;
	c1=c2;
	if (n1 = = n2 && c1 = = c2)
		6 < 9 and N=N

	print ("%d = %d and %c = %c" , n1, n2, c1, c1);
	else
	if (n1 < n2 && c1 = = c2)
	printf ("%d < %d and %c = %c", n1, n2, c1, c2);
	else
	printf ("Better Luck Next Time !");	

(d)	int a = 34, b = 32, c = 7, d = 15;
	a = b + c + d;
	if (a < 100)
	a = a * 2;
	b = b * c;
	c = c + d;
	if (a >b && c = = d)
	{
	c = d
	b = c;
	a = b;
	}
	else
	if (a > b && c > d || b>=d+c)
	{
	d = c * c;
	a = b * b;
	}
	printf ("a = %d, b=%d, c=%d, d=%d", a, b, c, d);
		a = 50176, b = 224, c = 22, d = 484

e)	int x = 5, y = 7 z = 9;
	if (x % 2 = = 0)
	x ++;
	else
	x = y + z;
	printf (" x = %d\n", x);
	if (x % 2 = = 1 && y % 2 = = 1 && z % 2 = = 1)
	printf ("All are Odd");
	if (x > y || x < z)
	{
	if (x > y)
	y ++;
	else
	if (x < z)
	print ("x = %d, y = %d, z = %d" , x, y, z);
		x = 16
		x = 16, y = 8, z = 9

PROGRAMMING EXERCISES (A.B)

EXERCISE 1

Write a program that takes two integers as input and tells whether first one is a factor of the second one?

```
# include <stdio.h>
        void main( )
        {
            int num_1, num_2;
            printf("enter first integer ");
            scanf("%d", &num_1);
            printf("enter second integer ");
            scanf("%d", &num_2);
            if(num_2%num_1==0)
            printf("Yes first one is the factor of second one");
            else
            printf("No first one is not the factor of second one");
        }
```

EXERCISE 2

Write a program that takes a number as input and displays "YES" if the input number is multiple of 3, and has 5 in unit's place e.g. 15, 75.

```
# include <stdio.h>
            void  main  ( )
        {
            int num;
            printf("input a number that has 5 in unit place (e.g. 15,75) ");
            scanf("%d", &num);
            if(num%3==0)
            printf("yes the input number is a multiple of 3");
            else
            printf("No the input number is not a multiple of 3");
        }
```

EXERCISE 3

Following is the list of discounts available in "Grocery Mart".

Total Bill	Discount
1000	10%
2500	20%
5000	35%
10000	50%

Write a program that takes total bill as input and tells how much discount the user has got and what the discounted price is.

```
# include <stdio.h>
                    void main( )
              {
                    float total_bill , discount;
                    printf("Enter your Total bill.");
                    scanf("%f", & total_bill);
                    if(total_bill>=1000 && total_bill<2500)
                    {
                       discount=total_bill*10/100;
                    printf("discounted price=%f", discount);

                    }
                    else if(total_bill>=2500 && total_bill<5000)
                    {
                       discount=total_bill*20/100;
                       printf("discounted price=%f", discount);
                    }
                    else if(total_bill>=5000 && total_bill<10000)
                    {
                       discount=total_bill*35/100;
                       printf("discounted price=%f", discount);
                    }
                    else if(total_bill>=10000)
                    {
                       discount=total_bill*50/100;
                 printf("discounted price=%f", discount);
                    }
              }
```

EXERCISE 4

Write a program that takes as input, the original price and sale price of a product and tells whether the product is sold on profit or loss. The program should also tell the profit/loss percentage.

```
#include <stdio.h>
void main()
{
                    float cost_price,selling_price, amount;
                       float profit_per, loss_per;
                    printf("Enter cost price: ");
                    scanf("%f", &cost_price);
                       printf("Enter selling price: ");
                    scanf("%f", &selling_price);
                    if(selling_price > cost_price)
                    {
                    amount = selling_price - cost_price;
                    profit_per=((amount*100)/cost_price);
                    printf("Profit = %f", amount);
                    printf("\nProfit_prcentage = %f", profit_per);
```

```
                    }
                    else if(cost_price > selling_price)
                    {
                    amount = cost_price - selling_price;
                    loss_per=((amount*100)/cost_price);
                    printf("Loss = %f", amount);
                    printf("Loss_prcentage = %f", loss_per);
                    }
                    else
                    {
                    printf("No Profit No Loss.");
                    }
}
```

EXERCISE 5

Write a program that takes as input, the lengths of 3 sides of a triangle and tells whether it is a right-angle triangle or not. For a right-angled triangle,
Hypotenuse^2= base^2 + height^2

```
# include <stdio.h>
            void main ( )
      {
            float side1,side2,side3;
            printf("Enter three sides of a triangle ");
            scanf("%f%f%f", &side1, &side2, &side3);
if(side1*side1 == side2*side2+side3*side3|| side2*side2 == side1*side1+side3*side3||
side3*side3==side1*side1+side2*side2)
                    printf("yes");

            else
                    printf("No");
      }
```

EXERCISE 6

Following is the eligibility criteria for admission in an IT University.

- **At least 60% marks in Matric.**
- **At least 65% marks in Intermediate (Pre-Engineering or ICS)**
- **At least 60% marks in entrance test**

Write a program that takes as input, the obtained and total marks of Matric, Intermediate and Entrance Test. The program should tell whether the students is eligible or not.

```
# include <stdio.h>
            void main ( )
            {
                    float obt_marks_matric, total_marks_matric;
                    float obt_marks_inter, total_marks_inter;
                    float obt_marks_entry_test, total_marks_entry_test;
                    float matric_prc, inter_prc, entry_test_prc;
                    printf("Enter Total marks of matric exams");
                    scanf("%f", &total_marks_matric);
                    printf("Enter obtained marks of matric exams");
                    scanf("%f", &obt_marks_matric);
                    printf("Enter Total marks of Intermediate exams");
                    scanf("%f", &total_marks_inter);
```

```
                printf("Enter obtained marks of Intermediate exams");
                scanf("%f", &  obt_marks_inter);
                printf("Enter Total marks of Entrance Test");
                scanf("%f", & total_marks_entry_test);
                printf("Enter obtained marks of Entrance Test exams");
                scanf("%f", & obt_marks_entry_test);
                matric_prc= obt_marks_matric/total_marks_matric*100;
                inter_prc= obt_marks_inter/total_marks_inter*100;
                entry_test_prc= obt_marks_entry_test/total_marks_entry_test*100;
                if(matric_prc>=60 && inter_prc>=65 && entry_test_prc>=65)
                printf("You are eligible for admission");
                else
                printf("You are not eligible for admission");
        }
```

EXERCISE 7

Write a program that calculates the bonus an employee can get on the following basis:

Salary	Experience with Company	Bonus Tasks	Bonus
10000	2 year	5	1500
10000	3 year	10	2500
25000	3 year	4	2000
75000	4 year	7	3500
100000	5 year	10	5000

The program should take as input, experience and number of bonus tasks of the employee. The program should display the bonus on the screen.

```
# include <stdio.h>
                void main ( )
                {
                        int salary, experience, bonus_tasks;
                        printf("Enter your current salary.");
                        scanf("%d", & salary);
                        printf("enter your experience with company in years.");
                        scanf("%d", & experience);
                        printf("Enter number of bonus tasks of the employee.");
                        scanf("%d", & bonus_tasks);
                        if(salary==10000 && experience==2 && bonus_tasks==5)
                        printf("Your calculated bonus = 1500");
                        else if(salary==10000 && experience==3 && bonus_tasks==10)
                        printf("Your calculated bonus =2500");
                        else if(salary==25000 && experience==3 && bonus_tasks==4)
                        printf("Your calculated bonus =2000");
                        else if(salary==75000 && experience==4 && bonus_tasks==7)
                        printf("Your calculated bonus =3500");
                        else if(salary==100000 && experience==5 && bonus_tasks==10)
                        printf("Your calculated bonus =5000");
                        else
                        printf("Invalid Input");
                }
```

ANSWER KEY

3.1 CONTROL STATEMENT

1	2	3	4	5
A	A	B	B	D

3.2 SELECTION STATEMENT

3.2.1 IF STRUCTURE

3.2.2 IF-ELSE STRUCTURE

3.2.3 NESTED SELECTION STRUCTURES

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	A	B	B	D	A	A	A	A	D	A	B	A	B	B	D

CH # 4 DATA AND REPETITION

4.1 DATA STRUCTURE

SHORT QUESTIONS

Q.1 Define Data Structure. (K.B)

Ans: Data structure is a container to store collection of data items in a specific layout.

Q.2 How many forms do data structure have? (K.B)

Ans: There are generally four forms of data Structures:

- Linear: arrays, lists.

MULTIPLE CHOICE QUESTIONS

1. _____ structure is a container to store data items. (K.B)

(A) Data (B) Control (C) Selection (D) Loop

2. Data structure is a container to store data in _____ layout. (K.B)

(A) Specific (B) Irregular (C) Alternative (D) None of these

3. _____ is type of data structure: (K.B)

(A) Array (B) List (C) Both A & B (D) None of these

4.1.1 ARRAY
4.1.2 ARRAY DECORATION
4.1.3 ARRAY INITIALIZATION
4.1.4 ACCESSING ARRAY ELEMENTS
4.1.5 USING VARIABLES AS ARRAY INDEXES.

LONG QUESTION

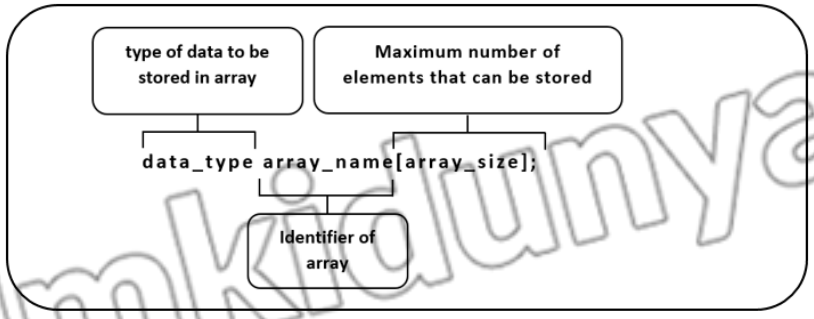
1. Briefly Explain Array in detail. (K.B+U.B)

Ans: **Definition**

An array is a data structure that can hold multiple values of same data type e.g. an 'int' array can hold multiple integer values, a 'float' array can hold multiple real values and so on. An important property of array is that it stores all the values at consecutive locations inside the computer memory.

Array Declaration:

In C language, an array can be declared as follows:



Array Initialization:

Assigning values to an array for the first time, is called array initialization. An array can be initialized at the time of its declaration, or later. Array initialization at the time of declaration can be done in the following manner.

Data_type array_name[N] = {value1, value2, value3,……, value N};

Char vowels [5] = {'a', 'e', 'i', 'o', 'u'}

If we do not initialize an array at the time of declaration. Then we need to initialize the array elements one by one. It means that we cannot initialize all the elements of array in a single statement.

Elements of Array:

Each element of an array has an index that can be used with the array name as array-name [index] to access the data stored at that particular index.
First element has the index 0, second element has the index 1 and so on. Thus height [0] refers to the first element of array height, height [1] refers to the second element and so on. Figure shows graphical representation of array height initialized in the last section.

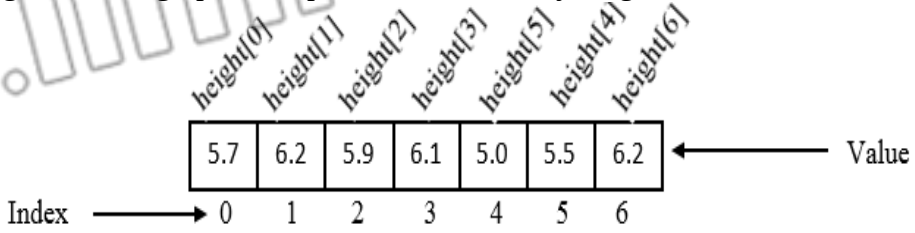


Figure 4.1: Graphical representation of array height

Example:
Write a program that stores the ages of five persons in an array, and then displays on screen.
Solution:

```
#include<stdio.h>
void main()
{
int age [5];
/* Following statements assign values at different indices of array age, we can see that the first value is stored at index 0 and the last value is stored at index 4 */
age [0] = 25;
age [1] = 34;
age [2] = 29;
age [3] = 43;
age [4] = 19;
/* Following statement displays the ages of five persons stored in the array *\
printf("The ages of five persons are: %d, %d, %d, %d, %d", age [0], age [1], age [2], age [3], age [4]);
```

SHORT QUESTIONS

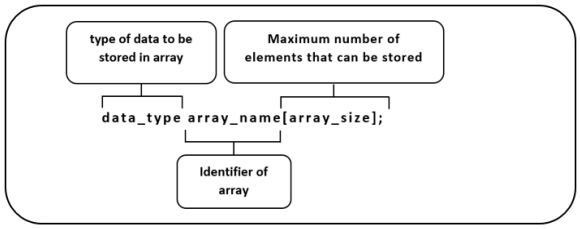
Q.1 Define Array. (K.B)

Ans: An array is a data structure that can hold multiple values of same data type e.g. an 'int' array can hold multiple integer values, a 'float' array can hold multiple real values and so on. An important property of array is that it stores all the values at consecutive locations inside the computer memory.

Data_type array_name[N] = {value1, value2, value3,……, value N};

Q.2 Draw a diagram to show array declaration. (K.B+U.B+A.B)

Ans:



Q.3 Write down the syntax to initialize an Array. (U.B+A.B)

Ans: *Data_type array_name[N] = {value1, value2, value3,……, value N};*

Q.4 Define Array Initialization. (K.B)

Ans: Assigning values to an array for the first time, is called array initialization. An array can be initialized at the time of its declaration, or later. Array initialization at the time of declaration can be done in the following manner.

Data_type array_name[N] = {value1, value2, value3,……, value N};

Q.5 Define Element of an Array. (K.B)

Ans: Each element of an array has an index that can be used with the array name as array-name[index] to access the data stored at that particular index.

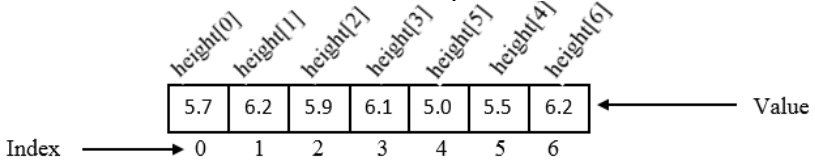


Figure 4.1: Graphical representation of array height

MULTIPLE CHOICE QUESTIONS

1. Each element of array has _____. (K.B)

(A) Value (B) Index (C) Subscript (D) Both B & C

2. _____ is the by default index. (K.B)

(A) 0 (B) 1 (C) 2 (D) Any

3. _____ can be used as any indexes. (K.B+U.B)

(A) Variable (B) Constant (C) Both A & B (D) None of these

4. Index can be _____.

(A) Variable (B) Constant (C) Both A & B (D) None of these

4.2 LOOP STRUCTURE
4.2.2 FOR LOOP
4.2.3 NESTED LOOP
4.2.5 LOOPS AND ARRAYS

LONG QUESTION

1. Define Loop. Explain *for* Loop in detail. (K.B+U.B)

Ans: Definition

"Loop is a control structure that repeats a statement or set of statements up to a fix number of times or until a specific condition is satisfied**"**

Types of loops:

'C' language provides three kind of loop structure:

1. *for* loop
2. *while* loop
3. *do while* loop

***for* Loop:**

Definition:

"*for"* loop is a type of loop which keeps on repeating a statement or a set of statements up to a fixed number of times**"**

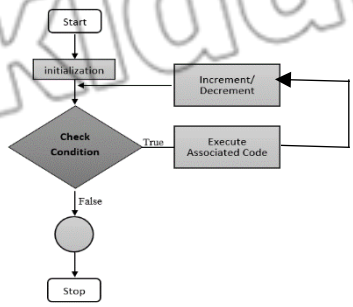
General Syntax:

In C programming language, ***for*** loop has the following general syntax.

```
for(variable_initialization; condition; increment/decrement)
{
Code to repeat
}
```

Flowchart:

In order to understand the ***for*** loop structure let's look at the following flow chart.



From the flow chart, we can observe the following sequence:

Step 1. Initialization is the first part to be executed in a ***for*** loop. Here we initialize our counter variable and then move to the condition part.

Step 2. Condition is checked, and if it turns out to be false, then we come out of loop.

Step 3. If the condition is true, then body of the loop is executed.

Step 4. After executing the body of loop, the counter variable is increased or decreased depending on the used logic and then we move again to the **step 2**.

Example:

```
for(int i = 1; i < = 10; i++)
{
printf("%d\n", i);
}
```

Output:

1
2
3
4
5
6
7
8
9
10

4.2.5 LOOPS AND ARRAYS

LONG QUESTIONS

Q.1 Relate Loops with Arrays. Also discuss how loop can be used to read and write values in arrays?

As variables can be used as array indexes, so we can use loops to perform different operations on array. If we want to display the whole array, then instead of writing all the elements on by one, we can loop over the array elements by using the loop counter as array index.

In the following, we discuss how loops can be used to read and write values in arrays.

1. Writing values in Array using Loops:

Using loops, we can easily take input in arrays. If we want to take input from user in an array of size 10, we can simply use a loop as follows:

Example:

```
int a [10];
for (int i =0, i < 10; i++)
scanf("%d", &a[i]);
```

2. Reading values from Arrays using Loops:

Loops help us in reading the values from array. The following code can be used to display the elements for an array having 100 elements:

Example

```
for (int i = 0; i < 100; i++)
printf("%d", a[i]);
```

SHORT QUESTIONS

Q.1 Define Loop. (K.B)

Ans: **"**Loop is a control structure that repeats a statement or set of statements up to a fix number of times of until a specific condition is satisfied**"**

Q.2 Write the names of different types of loops. (K.B)

Ans: C language provides three kind of loop structure:

1. *for* loop
2. *while* loop
3. *do while* loop

Q.3 Define *'for'* Loop. (K.B)

Ans: **"*for*"** loop is a type of loop which keeps on repeating a statement or a set of statements up to a fixed number of times**"**

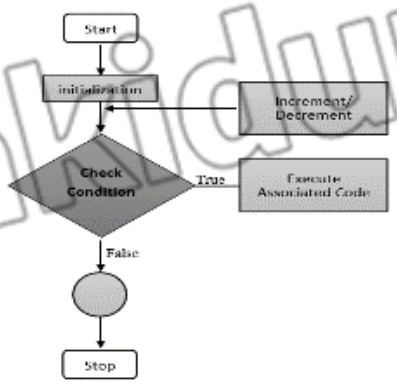
Q.4 Write down the syntax of '*for'* loop. (K.B+U.B)

Ans: In C programming language, '*for'* loop has the following general syntax.

```
for (initialization; condition; increment/decrement)
{
Code to repeat
}
```

Q.5 Draw flowchart to explain '*for'* loop. (K.B+U.B+A.B)

Ans:



Q.6 Write a program to print "Pakistan" three times using '*for'* loop. (A.B)

Ans:

```
for(int i = 0; i < 3; i++)
{
```

```
    printf("Pakistan\n");
}
```

Q.7 Write a program to display the values from 1 to 10 on screen. (A.B)

Ans:

```
for (int i = 1; i < = 10; i++)
{
    printf("%d\n", i);
}
```

Q.8 Write a program that displays the table of 2. (A.B)

Ans:

```
#include <stdio.h>
void main()
{
int n=2,j;
for(j=1;j<=10;j++)
{
      printf("%d X %d = %d \n",n,j,n*j);
}
}
```

Q.9 What will happen if condition don't return *false* in any case of '*for*' loop? (K.B+U.B)

Ans: If the condition in loop do not gets false at some point then the loop repeats for infinity and never terminates

Q.10 Define iteration. (K.B)

Ans: Each run of loop is called an iteration

Q.11 Define Nested '*for*' Loop. (K.B)

Ans: A loop in a loop is known as nested loop.

Q.12 Write down the structure of Nested '*for*' loop. (K.B)

Ans:

```
for (initialization; condition; increment/decrement)
{
for (initialization; condition; increment/decrement)
{
    Code to repeat
}
}
```

Q.13 Write a program that shows the working of nested '*for*' loop. (A.B)

Ans:

```
#include<stdio.h>
void main()
for(int i =1; i < = 5; i++)
{
for(int j =1; j < = i; j++)
{
printf("*");
}
printf("\n");
```

```
}
}
```

Q.14 Write a program that displays the table of 2, 3, 4, 5 and 6. (A.B)

Ans:

```
#include <stdio.h>
        void main()
{
        int n=2,j;
        printf("TABLE OF 2");
        for(j=1;j<=10;j++)
  {
        printf("%d x %d = %d \n",n,j,n*j);
  }
        n=3;
        printf("\nTABLE OF 3");
        for(j=1;j<=10;j++)
  {
        printf("%d x %d = %d \n",n,j,n*j);
  }
        n=4;
        printf("\nTABLE OF 4");
        for(j=1;j<=10;j++)
  {
        printf("%d x %d = %d \n",n,j,n*j);
  }
        n=5;
        printf("\nTABLE OF 5");
        for(j=1;j<=10;j++)
  {
        printf("%d x %d = %d \n",n,j,n*j);
  }
        n=6;
        printf("\nTABLE OF 6");
        for(j=1;j<=10;j++)
  {
        printf("%d x %d = %d \n",n,j,n*j);
  }
}
```

Q.15 How loops can be used to read and write values in arrays. (K.B+U.B)

Ans: Writing values in Array using Loops:

Using loops, we can easily take input in arrays. If we want to take input from user in an array of size 10, we can simply use a loop as follows:

```
int a [10];
```

```
for (int i =0, i < 10; i++)
scanf("%d", &a[i]);
```

Reading values from Arrays using Loops: Loops help us in reading the values from array. The following code can be used to display the elements for an array having 100elements:

```
for (int i = 0; i < 100; i++)
printf("%d", a[i]);
```

MULTIPLE CHOICE QUESTIONS

1. A structure that enables the programmer to execute same sequence of statement repeatedly until a particular condition is met is called _____. (K.B)

(A) Loop (B) Sequence (C) Selection (D) Switch Statement

2. A loop has essential elements _____. (K.B+U.B)

(A) Two (B) Three (C) Four (D) Five

3. A loop terminates based on _____. (K.B+U.B)

(A) Test Condition (B) Block of Statement
(C) Increment (D) Decrement

4. A statement that is executed fixed number of times is called _____. (K.B)

(A) for Statement (B) while Statement
(C) do–while Statement (D) Nested loop Statement

5. A loop that is known as counter loop _____. (K.B)

(A) For (B) While (C) Do – While (D) Nested

6. When a *for Loop* is executed a variable is assigned on initial value in the loops called _____. (K.B+U.B)

(A) Loop variable (B) Variable (C) Increment (D) Decrement

7. The general syntax of *for* loop is _____. (K.B+U.B)

(A) for (initialization; test condition; increment / decrement)
(B) for (test condition; increment)
(C) for (initialization; test condition; post fix)
(D) for (variable; condition; and decrement)

8. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main()
{
int k =0;
for (k)
printf("Hello");
}
```

(A) Compile Time Error (B) Hello
(C) Nothing (D) Varies

9. What will be the output of following C code? (K.B+U.B+A.B)

```
#include <stdio.h>
void main()
{
int k;
for (k = -3 ; k < -5; k++)
printf("Hello");
}
```

(A) Hello (B) Infinite Hello (C) Run Time Error (D) Nothing

10. What will be the output of following C code? (K.B+U.B+A.B)

```
for (i=0; i<10; i++)
```

printf("%d", i);

(A) 10 (B) 0123456789 (C) 0 (D) Syntax Error

11. What will be the output of following C code? (K.B+U.B+A.B)

for(i=0; i++; i<15)

printf("%d ", i);

(A) 10 11 12 13 14 (B) 9 10 11 12 13

(C) Infinite Loop (D) 10 11 12 13 14 15 16

12. What will be the final value of the variable 'digit'? (U.B+A.B)

```
void main()
{
int digit = 0;
for( ; digit < = 9; )
digit++;
digit *= 2;
digit;
}
```

(A) -1 (B) 17 (C) 19 (D) 26

13. Which one of the following is not a loop statement? (K.B)

(A) for (B) if (C) while (D) do – while

14. When number of iterations of execution are known in advance, which is the best choice of programmer? (K.B+U.B)

(A) for (B) do – while (C) while (D) None of These

15. How many expressions are used in *for* Loop? (K.B)

(A) 2 (B) 4 (C) 5 (D) 3

16. Which are the mandatory part of *for* Loop? (K.B+U.B)

(A) Initialization (B) Condition

(C) Increment/Decrement (D) All of these

17. The *for* loop expressions are enclosed in: (K.B+U.B)

(A) ( ) (B) { } (C) [ ] (D) None of These

18. Which of the following expression is executed only once in the *for* Loop? (K.B+U.B)

(A) Condition Loop Control (B) Increment/Decrement
(C) Initialization (D) None of These

19. *for* loop is also called: (K.B)

(A) Conditional Loop (B) Counter Loop (C) Sentinel Loop (D) Nested Loop

20. The expressions in the *for* loop is separated by: (K.B+U.B)

(A) , (B) ; (C) ¦ (D) .

21. Which part of loop statement is used to control the loop? (K.B+U.B)

(A) Body (B) Increment /Decrement

(C) Condition (D) None of These

22. The loop which never end is called: (K.B+U.B)

(A) Running Loop (B) Infinite Loop (C) Nested Loop (D) Continuous Loop

23. Which one of the following is loop statement? (K.B+U.B)

(A) if (B) if-else (C) switch (D) None of These

24. One execution of a loop is known as: (K.B)

(A) Cycle (B) Duration (C) Iteration (D) Both A & C

25. In *for* loop, this expression is executed only once: (K.B+U.B)

(A) Test (B) Validation (C) Initialization (D) None of these

26. -------- is known as repetition structure. (K.B+U.B)

(A) Loop (B) Selection (C) Sequential (D) None of these

27. --------- repeats one or more statement. (K.B+U.B)

(A) Loop (B) Selection (C) Sequential (D) None of these

28. There are ---- types of loop. (K.B)

(A) 1 (B) 2 (C) 3 (D) 4

29. -------- loop is used to repeat a statement to a fix number of times. (K.B+U.B)

(A) for (B) Do (C) do-white (D) None of these

30. ------- is a first part to be executed in *for* loop. (K.B+U.B)

(A) Initialization (B) condition (C) increment (D) None of these

31. -------- is represented by 1

(A) True (B) False (C) Both A & B (D) None of these

32. Loop in a loop is called _____. (K.B+U.B)

(A) nested loop (B) for loop (C) do loop (D) None of these

33. Each run of a loop is called _____. (K.B)

(A) cycle (B) iteration (C) Both A & B (D) None of these

PROGRAMMING TIME (A.B

Programming Time 4.1

Write a program that stores the ages of five persons in an array, and then displays on screen.

Solution:

```
#include<stdio.h>
void main ()
{
        int age[5];
        /* Following statements assign values at different indices of array age.We can
        see that the first value is stored at index 0 and the last value is stored at index
        4 */
                age[0]   = 25;
                age[1]   = 34;
                age[2]   = 29;
                age[3]   = 43;
                age[4]   = 19;
/* Following statement displays the ages of five persons stored in the array  * /
.
printf("The ages of five persons are: %d, %d, %d, %d, %d", age[0], age[1], age[2],
                age[3], age[4]);
```

Programming Time 4.2

Write a program that takes the marks obtained in 4 subjects as input from the user, calculates the total marks and displays on screen.

Solution:

```
#include<stdio.h>
void main()
{
        float marks[4], total_marks;
        printf("Please enter the marks obtained in 4 subjects:")
        scanf("%f%f%f%f, &marks[0], &marks[1], &marks[2],&marks[3]);
        total_marks  = marks[0]+marks[1]+marks[2]+marks[3];
        printf("Total  marks  obtained  by· student  are  %f",total_marks);
}
```

Programming Time 4.3

Write a program that displays the values from 1 -- 10 on the computer screen.

Program:

```
for(int i=1;i<=10;i++)
{
        printf("%d\n",i);
```

}
Description:
Consider the example program given above.

- First of all, the value of i is set to 1 and then condition is checked.
- As the condition is true (1<= 10) so loop body executes. As in the loop body, we are displaying the value of the counter variable, so 1 is displayed on console.
- After increment, the value of i becomes 2. The condition is again checked. It is true as(2<=10) so this time 2 is printed.
- The procedure continues till 10 is displayed and after increment the value of i becomes 11. Condition is checked and it turns out to be false (11>10) so the loop finally terminates after printing the numbers from 1 to 10.

Programming Time 4.4

Write a program that calculates the factorial of a number input by user.
Program Logic:
When we want to solve a problem programmatically, first we need to know exactly what we want to achieve. In this example, we are required to find the factorial of a given number, so first we need to know the formula to find factorial of a number.

N! = 1 * 2 * 3 * 4 * ..... * (N-1) * N

We can see the pattern that is being repeated, so we can solve the problem using for loop.
Program:

```
#include<stdio.h>
void main()
{
        int n, fact = 1;
        printf("Please enter a positive number whose factorial you want to find");
        scanf("%d, &n);
        for(int I = 1; i <=n; i++)
        {
                fact = fact * i;
        }
        printf("The factorial of input number %d is %d, n, fact);
}
```

Description:
Following table shows the working of program, if the input number is 5. It demonstrates the changes in the values of variables at each iteration.

Iteration	Value of counter	Condition	Loop body	Result
				fact=1
1	i=1	TRUE(1<=5)	fact = fact*i	fact=1*1=1.
2	i=2	TRUE(2<=5)	fact = fact*i	fact=1*2=2
3	i=3	TRUE(3<=5)	fact = fact*i	fact=2*3=6

4	i=4	TRUE(4<=5)	fact = fact*i	fact=6*4=24
5	i=5	TRUE(5<=5)	fact = fact*i	fact=24*5=120
6	i=6	FALSE(6>5)		

Programming Time 4.5

Problem:
Write a program that 5 times displays the numbers from 1-10 on computer screen.
Program:

```
#include<stdio.h>
void main()
{
        for(int i = 1; i <= 5; i++)
        {
                for(int j = 1; j <=10; j++)
                {
                        printf(%d", j);
                }
                printf("\n");
        }
```

}

Output:

Here is the output of above program.

```
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
```

Description:

As we understand the working of inner loop, so here let's focus on outer loop.

1- For the value i = 1, condition in outer loop is checked which is true (1 < = 5), so whole inner loop is executed and numbers from 1 – 10 are displayed.
2- When control gets out of inner loop, printf("\n"); is executed which inserts a new line on console.
3- Then i is incremented and it becomes 2. As it is less than 5, so condition is true. The whole inner loop is executed, and thus numbers from 1 – 10 are again displayed on screen. Coming out of the inner loop new line is inserted again.
4- After five times displaying the numbers from 1 – 10 on screen, the value of i gets incremented to 6 and condition of outer loop turns false. So outer loop also terminates.

Programming Time 4.6

Problem:

Write a program to display the following pattern of stars on screen.

```
*
**
***
****
*****
******
```

Program:

```
#include<stdio.h>
void main ( )
{
        for(int i = 1; i <= 6; i++)
        {
                for(int j = 1; j <= i; j++)
                        printf("*");
                printf("\n");
        }
}
```

Description:

Here is the description of above code.

1- As we have to display 6 lines containing stars, so we run the outer loop from 1 to 6.
2- We can observe that in the given pattern we have 1 star on 1st line, 2 stars on 2nd line, 3 stars on 3rd line and so on. So, the inner loop is dependent on the outer loop, i.e. if counter of outer loop is 1 then inner loop should run 1 time,

if the counter of outer loop is 2 then inner loop should run 2 times and so on. So, we use the counter of outer loop in the termination condition of inner loop i.e. j <= i.

3- When outer loop counter i has value 1, inner loop only runs 1 time, so only 1 star is displayed. When outer loop counter is 2, the inner loop runs 2 times, so 2 stars are displayed and the process is repeated until six lines are complete.

Programming Time 4.7

Problem:

Write a program that counts multiples of a given number lying between two numbers.

Program:

```
#include <stdio.h>
void main ()
{
        int n, lower, upper, count = 0;
        printf (“Enter the number:  ”);
        scanf (“%d", &n);
        printf (“Enter the lower and upper limit of multiples:\n");
        scanf (“%d%d”, &lower, &upper);
        for(int i = lower; i <= upper; i++)
                if(i%n == 0)
                        count++;
        printf ("Number of multiples of %d between %d and %d are %d”, n, lower,
        upper, count);
}
```

Programming Time 4.8

Problem:

Write a program to find even numbers in integers ranging from n1 to n2 (where nl is greater than n2).

Program:

```
#include  <stdio.h>
void main ( )
{
        int n1, n2;
        printf (“Enter the lower and upper limit of even numbers: \n”)
        scanf (“%d%d", &n2, &n1);
        if(n1 > n2)
        {
                for (int i = n1; i > = n2; i--)
                {
```

```
                    if(i % 2 == 0)
                        printf ("%d", i);
                }
        }
}
```

Programming Time 4.9

Problem:

Write a program to determine whether a given number is prime number or not.

Program:

```
#include <stdio.h>
void main ( )
{
        int n;
        int flag = 1;
        printf ("Enter a number:  ");
        scanf ("%d", &n);
        for (int i = 2; i < n; i++)
        {
                if (n % i == 0)
                        flag = 0;
        }
        if (flag == 1)
                printf ("This is a prime number");
        else
                printf ("This is not a prime number");
}
```

Programming Time 4.10

Problem:

Write a program to display prime numbers ranging from 2 to 100.

Program:

```
# include<stdio.h>
int main ()
{
        int flag;
        for (int j = 2; j <= 100;  j++)
        {
                flag = 1;
```

```
                for (int i = 2; i < j; i++)
                {
                        if(j % i == 0)
                        {
                                flag = 0;
                        }
                }
                if (flag == 1)
                {
                        printf ("%d", j);
                }
        }
}
```

Programming Time 4.11

Problem:
Write a program that assigns first 5 multiples of 23 to an array of size 5.
Program:

```
#include<stdio.h>
void main( )
{
        int multiples [5];
        for (int i = 0; i < 5; i++)
                multiples [i] = (i + 1) * 23;
}
```

Programming Time 4.12

Problem:
Write a program that adds corresponding elements of two arrays.
Program:

```
#include <stdio.h>
void main ( )
{
        int a[ ] = {2, 3, 54, 22, 67, 34, 29, 19);
        int b[ ] = {65,73, 26, 10, 4, 2, 84, 26);
        for (int i=0; i<8; i++)
                printf ("%d ", a[i] +b[i]);
}
```

SOLVED ACTIVITIES (A.B

ACTIVITY 4.1

Write a program that displays the table of '2'.
Solution:

```
#include <stdio.h>
void main()
{
int n=2,j;
for(j=1;j<=10;j++)
```

```
{
printf("%d X %d = %d \n",n,j,n*j);
}
}
```

ACTIVITY 4.2

Write a program that displays the table of 2,3,4,5 and 6.

Solution:

```
#include <stdio.h>
void main()
{
int n=2,j;
for(j=1;j<=10;j++)
{
printf("%d x %d = %d \n",n,j,n*j);
}
n=3;
for(j=1;j<=10;j++)
{
printf("%d x %d = %d \n",n,j,n*j);
}
n=4;
for(j=1;j<=10;j++)
{
printf("%d x %d = %d \n",n,j,n*j);
}
n=5;
for(j=1;j<=10;j++)
{
printf("%d x %d = %d \n",n,j,n*j);
}
n=6;
for(j=1;j<=10;j++)
{
printf("%d x %d = %d \n",n,j,n*j);
}
}
```

ACTIVITY 4.3

Write a program that takes as input the marks obtained in matriculation by 30 students of a class. The program should display the average marks of the class.

Solution:

```
#include<stdio.h>
void main()
{
```

```
int i;
int Avg;
int maks[30];                          //Array Declaration
int sum=0;
for( i=0 ; i<=29 ; i++)
{
printf(" Enter Marks/n ");
scanf("%d" , &marks[i] );              // Store Data in      Array
}
for( i=0 ; i<=29 ; i++ )
{
Sum = sum + marks[i];                  // Read Data  from an Array
Avg = Sum/30;
printf(" Average Marks = %d\n" , Avg);
}
}
```


EXERCISE

Q1. Multiple Choice Questions

1) An array is a_____structure. **(K.B)**
A) Loop B) Control C) Data D) Conditional

2) Array elements are stored at _____ memory locations. **(K.B)**
A) Contiguous B) Scattered C) Divided D) None

3) If the size of an array is 100, the range of indexes will be _____. **(K.B)**
A) 0-99 B) 0-100 C) 1-100 D) 2-102

4) _____ structure allows repetition of a set of instructions. **(K.B+U.B)**
A) Loop B) Conditional C) Control D) Data

5) _____ is the unique identifier, used to refer to the array. **(K.B)**
A) Data Type B) Array Name C) Array Size D) None

6) Array can be initialized _____ declaration. **(K.B)**

A) At the time of B) After C) Before D) Both a & b

7) Using loops inside loops is called _____loops. (K.B+U.B)

A) For B) While C) Do-while D) Nested

8) _____ part of for loop is executed first. (K.B)

A) Condition B) Body

C) Initialization D) Increment/Decrement

9) _____ make it easier to read and write values in array. (K.B+U.B)

A) Loops B) Conditions C) Expressions D) Functions

10) To initialize the array in a single statement, initialize it_ declaration. (K.B+U.B)

A) At the time of B) After C) Before D) Both A & B

ANSWER KEY

1	2	3	4	5	6	7	8	9	10
C	A	A	A	B	D	D	C	A	A

Q2. Define the following terms. (K.B)

1) Data Structure

Ans: Data structure is a container to store collection of data items in a specific layout. Array is the most commonly used Data Structure of C language.

2) Array

Ans: An array is a data structure that can hold multiple values of same data type e.g. an int array can hold multiple integer values, a float array can hold multiple real values and so on. An important property of array is that it stores all the values at consecutive locations inside the computer memory.

Syntax

Data_type array name [N] = {value1, value2, value3,……, value N};

3) Array Initialization

Ans: Assigning values to an array for the first time, is called array initialization. An array can be initialized at the time of its declaration, or later. Array initialization at the time of declaration can be done in the following manner.

Data_type array_name[N] = {value1, value2, value3,……, value N};

4) Loop Structure

Ans: If we need to repeat one or more statements, then we use loops. For example, if we need to write Pakistan thousand times on the screen then instead of writing printf("Pakistan"); a thousand times, we use loops. C language provides three kind of loop structure:

1. for loop
2. while loop
3. do-while loop

5) Nested Loops

Ans: A loop within a loop is known as nested loop.

General Structure

```
for(initialization; condition; increment/decrement)
{
for(initialization; condition; increment/decrement)
{
Code to repeat
}
}
```

Q3. Briefly answer the following questions. (K.B+U.B)

1) Is loop a data structure? Justify your answer.

Ans: No Loop is not a data structure. Loop is a type of control structure which repeats a statement of set of statements. While Data Structure is a container to store collection of data items in a specific layout.

2) What is the use of nested loops?

Ans: When we use a loop inside another loop, it is called nested loop structure. We use nested loops to repeat a pattern multiple time.

General Structure

```
for (initialization; condition; increment/decrement)
{
for (initialization; condition; increment/decrement)
{
Code to repeat
}
}
```

3) What is the advantage of initializing an array at the time of declaration?

Ans: If we do not initialize an array at the time of declaration. Then we need to initialize the array elements one by one. It means that we cannot initialize all the elements of array in a single statement. This is demonstrated by the following example.

Example:

```
void main()
{
        Int array [5];
        Array[5]= {10, 20, 30, 40, 50};
}
```

ERROR Initializing whole array after declaration not allowed

The compiler generates an error on the above example code, as we try to initialize the whole array in one separate statement after declaring it.

4) Describe the structure of a for loop.

Ans:
```
for (initialization; condition; increment/decrement)
{
Code to repeat
}
```

Step 1. Initialization is the first part to be executed in a for loop. Here we initialize our counter variable and then move to the condition part.

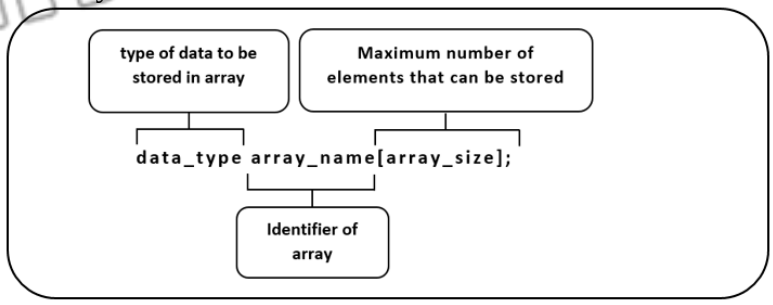
Step 2. Condition is declared and checked, and if it turns out to be false, then we come out of loop.

Step 3. If the condition is true, then body of the loop is executed.

Step 4. After executing the body of loop, the counter variable is increased or decreased depending on the used logic, and then we again move to the **step 2**.

5) How can you declare an array? Briefly describe the three parts of array declaration.

Ans: In C language, an array can be declared as follows:



1. Type of data to be stored in array.
2. How many maximum elements needed to be stored
3. Array name (known as identifier)

Q4. Identify the errors in the following code segments. (K.B+U.B+A.B)

Ans:

Sr #	Program	Error
A	int a [] = ({2},{3},{4});	Error expected in array initialization wrong syntax used for initializing array
B	for (int i = 0, i<10, i++) printf ("%d\", i);	Error expected in the syntax of for loop statement terminator is not used. for(int i=0,i<10,i++) **accurate syntax: for(int =0;i<10;i++)**

C	int a [] = {1,2,3,4,5};
	for (int j = 0; j<5; j++)
	printf("%d", a(j));
		Error expected in printf("%d",a(j));
		writing a variable in brackets is not a proper syntax.

D	float f [ ] = {1.4, 3.5, 7.3, 5.9};
	int size =4;
	for (int n = - 1; n< size; n--)
	printf("%f\n", f [n]);
		Error expected in printf("%f\n",f[n]);
		index value of an array cannot be negative.

E	int count = 0;
	for (int i =4; i<6; i--)
	for(int j = i, j<45; j++)
	{
	count++;
	pirntf("%count", count)
	}
		Error expected in printf("%count",count)..
		• %count is not right syntax of integer format specifier.
		• Statement terminator (;) missing.

Q5. Write down output of the following code segments. (K.B+U.B+A.B)

Ans:

Sr #	Program	Output

a	int sum = 0, p;
	for (p = 5; p < = 25; p = p +5)
	sum = sum + 5;
		Sum is 25

b	int i ; for (i = 34; i < = 60; i = i * 2) printf ("* ");	*

c	for (int i = 50; i < = 50; i ++)
	{
	for (j = i; j > = 48; j --)
	printf("j=%d \ n", j);
	printf ("i = % d \ n", i);
	}
		j = 50
		j = 49
		j = 48
		i = 50

d	int i, arr [ ] = {2, 3, 4, 5, 6, 7, 8};	4

	for (int = 0; i < 7; i++)
	{
	printf("%d\n", arr[i] * arr[i]);
	i++;
	}
		16
		36
		64

e	int i, j;
	float ar1[ ] = {1.1, 1.2, 1.3};
	float ar2[ ] = {2.1, 2.2, 2.3};
	for(i = 0; i < 3; i++)
	for (j = i; j<3; j++)
	printf("%f\n", ar1 [i] * ar2 [j] * i * j);
		0.000000
		0.000000
		0.000000
		2.640000
		5.520000
		11.959999

PROGRAMMING EXERCISES (A.B)

EXERCISE 1

Use loops to print following patterns on console.

a)
```
*****
*****
*****
```

b)
```
A
BC
DEF
GHIJ
```

KLMN

(A)

```
# include<stdio.h>
void main ( )
{
int i,j;
for(i=1;i<=3;i++)
{
for(j=1;j<=i;j++)
{
printf(“*****”);
}

printf(“\n”);

}

}
```

(B)

```
#include <stdio.h>
int main( )
{
int i,j,rows=5;
char alphabet='A';
printf("\nHere your pattern\n");
for(i=1; i<=rows; i++)
{
for(j=1; j<=i; j++)

{
printf(alphabet++);
}
printf(“\n”);

}

}
```

EXERCISE 2

Write a program that takes two positive integers a and b as input and displays the value of a^b.

```
# include <stdio.h>
int main ( )
        {
                int a,b,result=1;
                printf("Enter a:");
                scanf ("%d",&a);
                printf("Enter b:");
                scanf ("%d",&b);
                for (int i=1;i<=b;i++)
      {
         result=result*a;
      }
   printf("result=%d^%d=%d",a,b,result);
        }
```

EXERCISE 3

Write a program that takes two numbers as input and displays their Greatest Common Divisor (GCD) using Euclidean method.

```
# include <stdio.h>
void main( )
{
int m, n,r;
printf("Enter-two integer numbers: ");
scanf ("%d %d", &m, &n);
for (;n > 0;)
```

```
{
r = m % n;
m = n;
n = r;
}
printf ("GCD = %d \n",m);
}
```

EXERCISE 4

Write a program to display factorials numbers from 1 to 7. (Hint: Use Nested Loops)

```
# include<stdio.h>
void main ( )
{
int i,j,fact=1;
for(i=1;i<=7;i++)
{
for(j=1;j<=i;j++)
{
fact=fact*j;
}
printf("\n%d",fact);
fact=1;
}
}
```

EXERCISE 5

Write a program that takes 10 numbers as input in an array and displays the product of first and last element on console.

```
# include<stdio.h>
void main ( )
{
int i,arr[10],mult;
printf("Enter 10 elements:");
for(i=0;i<10;i++)
scanf("%d",&arr[i]);
mult=arr[0]*arr[9];
printf("Result of first and last element =%d",mult);
}
```

EXERCISE 6

Write a program that declares and initializes an array of 7 elements and tells how many elements in the array are greater than 10.

```
# include <stdio.h>
void main ( )
{
int counter=0,arr[ ]={3,54,22,67,34,29,19};
for(int i=0;i<7;i++)
{
if(arr[i] > 10)
```

```
counter++;
}
printf("The total number of elements greater than 10 are %d",counter);
}
```

ANSWER KEY

4.1 DATA STRUCTURE

1	2	3
A	A	C

4.1.1 ARRAY

4.1.2 ARRAY DECRORATION

4.1.3 ARRAY INITIALIZATION

4.1.4 ACCESSING ARRAY ELEMENTS

4.1.5 USING VARIABLES AS ARRAY INDEXES.

1	2	3	4
A	A	B	C

4.2 LOOP STRUCTURE

4.2.2 FOR LOOP

4.2.3 NESTED LOOP

4.2.5 ARRAY AND LOOPS

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	A	A	A	A	A	A	A	D	A	A	D	C	A
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
C	B	C	C	C	C	B	D	D	C	A	A	C	A	B
31	32	33												
A	A	C												

CH # 5

FUNCTIONS

5.1 FUNCTIONS

5.1.1 TYPES OF FUNCTIONS
5.1.2 ADVANTAGES OF FUNCTIONS
5.1.3 SIGNATURE OF A FUNCTION
5.1.4 DEFINING A FUNCTION

LONG QUESTIONS

1. Define Functions. Explain its different types. (K.B+U.B)

Ans: "A function is a block of statements which performs a particular task"

Example:

printf is function that is used to display anything on computer screen, ***scanf*** is another function that is used to take input from the user. Each program has a main function which performs the tasks programmed by the user. Similarly, we can write other functions and use them multiple times.

General Syntax of a Function:

A function definition has the following general structure.

return_type function_name (data_type var1, data_type var2,..,data_type varN)
{
Body of the function
}

Types of Functions:

There are basically two types of function:

1. Built-in Functions
2. User Defined Functions

Built-in Functions:

The functions which are available in 'C' Standard Library are called Built-in Functions. These functions perform commonly used mathematical calculations, string operations, input/ output operations etc. For example ***printf*** and ***scanf*** are built-in functions.

User Defined Functions:

The functions which are defined by a programmer are called user-defined functions.

2. Define functions. Also discuss the advantages of function. (K.B+U.B)

Ans: "A function is a block of statements which performs a particular task"

Advantages of Functions:

Functions provide us several advantages.

1. Reusability:

Functions provide reusability of code. It means that whenever we need to use the functionality provided by the function, we just call the functions. We do not need to write the same set of statements again and again.

2. Separation of tasks:

Functions allow us to separate the code of one task from the code of other tasks. If we have a problem in one function, then we do not need to check the whole program for removing the problem. We just need to focus at one single function.

3. Handling the complexity of the problem:

If we write the whole program as a single procedure, management of the program becomes difficult. Functions divide the program into smaller units, and thus reduce the complexity of the problem.

4. Readability:

Dividing the program into multiple functions, improves the readability of the program.

3. Define Functions. How can a function be defined? (K.B+U.B)

Ans: "A function is a block of statements which performs a particular task"

The function signature does not describe how the function performs the task assigned to it. Function definition does that.

General Syntax of a Function:

A function definition has the following general structure.

return_type function_name (data_type var1, data_type var2,..,data_type varN)
{
Body of the function
}

Body of the function is the set of statements which are executed in the function to perform the specified task.

We need to call a function, so that it performs the programmed task. Following is the general structure used to make a function call.

function_name (value1, value2,.., valueN);

There may be multiple ***return*** statement in a function but as soon as the first return statement is executed, the function call returns and further statements in the body of function are not executed. Return is a keyword that is used to return a value to the calling function. Output of the function is called its ***return value.*** A function cannot return more than one value. If we try to get more than one value then compiler gives an error. There may be multiple return statement in a function but as soon as the first return statement is executed, the function call returns and further statements in the body of function are not executed.

Example:

```
void show pangram ()
{
printf ("\nA quick brown fox jumps over the lazy dog. \n");
}
```

As the above function does not return anything thus return type of the function is void.

SHORT QUESTIONS

1. Define Divide and Conquer Rule. (K.B)

Ans: In divide and conquer rule the problem is decomposed into sub problems. Rather on concentrating the bigger problem as a whole we try to solve each sub problem separately. This leads to a simple solution.

2. Define Functions. (K.B)

Ans: A function is a block of statements that performs a particular task. e.g. ***printf*** is function that is used to display anything on computer screen, ***scanf*** is another function that is used to take input from the user. Each program has a main function which performs the tasks programmed by the user. Similarly, we can write other functions and use them multiple times.

3. Give general syntax of Functions. (K.B+U.B+A.B)

Ans: A function definition has the following general structure.

return_type function_name (data_type var1, data_type var2,..,data_type varN)
{
Body of the function
}

4. Write down the name of types of Functions. (K.B)

Ans: There are two types of function:
1. Built-in Functions **2.** User Defined Functions

5. Define Built-in Functions. (K.B)

Ans: The functions that are available in C standard library are called built-in Functions. These functions perform commonly used mathematical calculations, string operations, input/ output operations etc. For example, ***printf*** and ***scanf*** are built-in functions.

6. Define User- Define Functions. (K.B)

Ans: The functions that are defined by a programmer are called user-defined functions.

7. Define reusability. (K.B)

Ans: Functions provide reusability of code. It means that whenever we need to use the functionality provided by the function, we just call the functions. We do not need to write the same set of statements again and again.

8. What is separation of tasks? (K.B)

Ans: Functions allow us to separate the code of one task from the code of other tasks. If we have a problem in one function, then we do not need to check the whole program for removing the problem. We just need to focus at one single function.

9. How can a function handle the complexity of a program? (U.B)

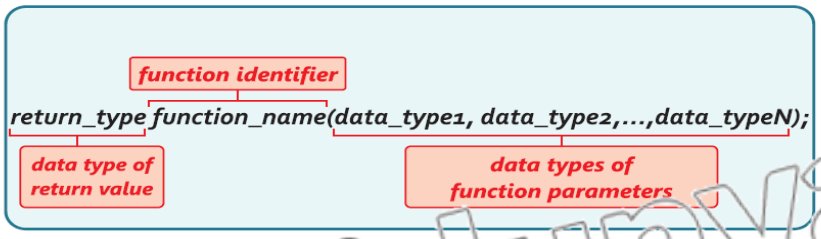
Ans: If we write the whole program as a single procedure, management of the program becomes difficult. Functions divide the program into smaller units, and thus reduce the complexity of the problem.

10. Define readability of a program. (K.B)

Ans: Dividing the program into multiple functions, improves the readability of the program.

11. What is signature of a function? (K.B)

Ans: A function is a block of statement that gets some inputs and provides some output. Inputs of a function are called ***parameters*** of the function and output of the function is called its ***return value.*** A function can have multiple parameters, but it cannot return more than one values. ***Function signature*** is used define the inputs and output of a function. The general structure of a function signature is as follows:



12. Show the descriptions of the following functions.

i. int square ii. float perimeter iii. int largest
iv. float area v. int isVowel

Ans:

Function Description	Function Signature
A function that takes an integer as input and returns its square.	int square (int);
A function that takes length and width of a rectangle as input and returns the perimeter of the rectangle	float perimeter (float , float);

A functions that takes three integers as input and returns the largest value among them.	int largest (int, int, int);
A function that takes radius of a circle as input and returns the area of circle.	float area (float);
A function that takes a character as input and returns 1, if the character is a vowel, otherwise returns 0.	int isvowel (char);

13. How can a function be called? (U.B)

Ans: Calling a function is the set of statements which are executed in the function to fulfil the specified task. Calling a function means to transfer the control to that particular function. During the function call, the values passed to the function are called ***arguments***. We can call a user-defined function from another user defined function, same as we call other functions in main function.

14. What is the purpose of the keyword 'return'? (K.B)

Ans: ***return*** is a keyword that is used to return a value to the calling function. Output of the function is called its **return value.** A function cannot return more than one value. If we try to get more than one value, then compiler gives an error. There may be multiple return statement in a function but as soon as the first return statement is executed, the function call returns and further statements in the body of function are not executed.

15. Define arguments. (K.B)

Ans: The values passed to the function are called ***arguments*.**

16. Define parameters. (K.B)

Ans: The variables in the function definition that receive these value is called ***parameters*** of the function.

17. Write down the difference between arguments and parameters. (K.B+U.B)

Ans: The values passed to the function are called ***arguments,*** whereas variables in the function definition that receive these value are called ***parameters*** of the function.

18. What points should be kept in mind for the arrangement of function? (K.B+U.B)

Ans: Following points must be kept in mind for the arrangement of function in a program.

1. If the definition of called function appears before the definition of called function, then function signature is not required.
2. If the definition of called function appears after the definition of calling function, then function signature of called function must be written before the definition of calling function.

MULTIPLE CHOICE QUESTIONS

1. Instead of solving the whole problem at once we try to apply _____ approach. (K.B+U.B)

(A) Divide & Conquer (B) Reusability
(C) Readability (D) None of these

2 _____ is a block of statements, which performs a particular task. (K.B)

(A) Program (B) Function (C) Both A & B (D) None of these

3. There are _____ type of function: (K.B)

(A) 1 (B) 2 (C) 3 (D) 4

4. _____ is an input function. (K.B)

(A) *printf* *(b) scanf* (C) Both A & B (D) None of these

5. _____ is an output function. (K.B)

(A) *printf* (B) *scanf* (C) Both A & B (D) None of these

6. Each program has _____ function. (K.B)

(A) *printf* (B) *scanf* (C) main (D) None of these

7. A function is a _____ of statements. (K.B)

(A) Block (B) Pattern (C) Standard (D) None of these

8. _____ are also known as library function. (K.B)

(A) Built in (B) User define function

(C) Both (D) None of these

9. Built in function is called_____ function. (K.B)

(A) Standard (B) Library (C) Predefined (D) All of these

10. Built in functions commonly performs _____. (K.B+U.B)

(A) Mathematical problems (B) String operations

(C) Input/ Output operation (D) All of these

11. _____ is type of built in function. (K.B)

(A) *printf* (B) *scanf* (C) Both A & B (D) None of these

12. User defined functions are defined by _____. (K.B)

(A) Programmer (B) Library (C) Developer (D) None of these

13. _____ functions are written by user. (K.B)

(A) User define (B) Built in (C) Both A & B (D) None of these

14. Functions provides _____ of code. (K.B+U.B)

(A) Reusability (B) Readability (C) Both A & B (D) None of these

15. We _____ need to write the code again and again in function. (K.B)

(A) Do (B) Do not (C) Can be both (D) None of these

16. We can increase the readability of program by _____ it. (K.B)

(A) Dividing (B) Multiplying (C) Reusing (D) None of these

17. Dividing a program increases _____ of program. (K.B+U.B)

(A) Reliability (B) Readability (C) Handling (D) None of these

18. Parameters are _____ of function. (K.B)

(A) Input (B) Output (C) Both A & B (D) None of these

19. ____ are inputs of function. (K.B)

(A) Parameters (B) Arguments (C) Both A & B (D) None of these

20. A function can have multiple _____. (K.B)

(A) Returns (B) Values (C) Parameters (D) None of these

21. A function can return _____ values. (K.B)

(A) One (B) Two (C) Three (D) Multiple

22. Function signature is used to defined_____. (K.B)

(A) Input (B) Output (C) Both A & B (D) None of these

23. int square(int); will _____. (K.B+U.B)
(A) Takes integer as input
(B) Takes the square of number
(C) Takes integer as input and return its square
(D) None of these

24. How many data types can be used while defining a function? (K.B+U.B)
(A) 1 (B) 2 (C) 3 (D) Multiple

25. What will be happen if we try to get more than one value from a function? (K.B+U.B)
(A) It will return multiple values (B) It will not return any value
(C) Compiler gives an error (D) None of these

26. How many return statements can be used with a function? (K.B)
(A) 1 (B) 2 (C) 3 (D) Multiple

27. When the first return statement is executed. (K.B)
(A) Further statement in body returns one by one
(B) Further statement stops executions
(C) Further statement in body executes in group
(D) None of these

28. Following is the general structure used to make a function call. (K.B)
(A) function _name (value1, value2…..valueN);
(B) function_name (value);
(C) function_name (value1, value2….valueN)
(D) None of these

29. _____ are the values passed to the functions. (K.B+U.B)
(A) Arguments (B) Parameters (C) Functions (D) None of these

30. Variables that receives the values in function defined are called _____.(K.B+U.B)
(A) Arguments (B) String case (C) Parameters (D) None of these

31. If the definition of function appears before the definition of called function then _____. (K.B+U.B)
(A) Signature function is required (B) Function signature is not required
(C) Error occurs (D) None of these

32. If the function of called function appears after the definition of calling function then _____. (K.B+U.B)
(A) Signature function is not required
(B) Function signature of called function must be written before the definition of calling function
(C) Error occurs
(D) None of these

33. The name of function should relate its _____. (K.B+U.B)
(A) Arguments (B) Parameters (C) Tasks (D) None of these

34. Calling a function means transfer the _____ to that Particular function. (K.B+U.B)
(A) Control (B) Path (C) Route (D) None of these

PROGRAMMING TIME (A.B)

Programming Time 5.1

Problem:
Write a function is Prime( ) that takes a number as input and returns 1 if the input number is prime, otherwise returns 0. Use this function in main( ).

Program:

```
#include <stdio.h>
int prime (int n)
{
        for (int i = 2; i < n; i++)
                if(n % i == 0)
                        return 0;
        return 1;
}
void main ( )
{
        int x;
        printf ("Please enter a number:  ");
        scanf ("%d", &x);
        if(prime(x))
                printf ("%d is a Prime Number", x);
        else
                printf ("%d is not a Prime Number",x);
}
```

Programming Time 5.2

Problem:
Write a function which takes a positive number as input and returns the sum of numbers from 0 to that number.

Program:

```
#include<stdio.h>
int digitsSum(int n)
{
        int sum=0;
        for(int i = 0; i <= n; i++)
        {
                sum = sum + i;
        }
        return sum;
}
void main( )
{
        int number;
        printf("Please enter a positive number:  ");
        scanf("%d", &number);
        if(number >= 0)
        {
                int sum = digitsSum(number);
                printf("The sum of numbers upto given number is %d", sum);
```

```
        }
        else
                printf("You entered a negative number.");
}
```

EXERCISE

Q1. Multiple Choice Questions

1. Function could be built-in or _____. (K.B)
A) Admin Defined B) Server Defined C) User Defined D) Both A & C

2. The functions which are available in C Standard Library are called_____. (K.B)
A) User-Defined B) Built-In C) Recursive D) Repetitive

3. The values passed to a function are called_____. (K.B+U.B)
A) Bodies B) Built-In C) Arrays D) Arguments

4. Char cd() {return 'a'}. in this function "char" is_____. (K.B)
A) Body B) Return Type C) Array D) Arguments

5. The advantages of using functions are_____. (K.B)
A) Readability B) Reusability C) Easy Debugging D) All

6. If there are three return statements in the function body, _____ of them will be executed. (K.B+U.B)
A) One B) Two C) Three D) First & Last

7. Readability helps to _____ the code. (K.B)
A) Understand B) Modify C) Debug D) All

8. _____ means to transfer the control to another function. (K.B)
A) Calling B) Defining C) Re-Writing D) Including

ANSWER KEY

1	2	3	4	5	6	7	8
C	B	D	B	D	A	D	A

Q2. Define the following terms. (K.B)

1) Functions.

Ans: A function is a block of statements that performs a particular task, e.g. ***printf*** is function that is used to display anything on computer screen, ***scanf*** is another function that is used to take input from the user. Each program has a main function which performs the tasks programmed by the user. Similarly, we can write other functions and use them multiple times.

2) Built-in functions.

Ans The functions which are available in C standard Library are called built-in Functions. These functions perform commonly used mathematical calculations, string operations, input/ output operations etc. For example, ***printf*** and ***scanf*** are built-in functions.

3) Functions Parameters.

Ans: A function is a block of statement that gets some inputs and provides some output. Inputs of a function are called **parameters** of the function. A function can have multiple parameters, but it cannot return more than one values.

4) Reusability.

Ans: Functions provide reusability of code. It means that whenever we need to use the functionality provided by the function, we just call the functions. We do not need to write

the same set of statements repeatedly.

5) Calling a function.

Ans: We need to call a function, so that it performs the programmed task. Following is the general structure used to make a function call.

Function_name (value1, value2,.., valueN);

Q3. Briefly answer the following questions. (K.B+U.B)

1) What is the difference between arguments and parameters? Give an example.

Ans: The values passed to the function are called **arguments,** whereas variables in the function definition that receive these value are called **parameters** of the function.

2) Enlist the parts of a function definition.

Ans: The function signature does not describe how the function performs the task assigned to it. Function definition does that. A function definition has the following general structure.

Return_type function_name (data_type var1, data_type var2,.., data_type VarN)
{
Body of the function
}

Body of the function is the set of statements which are executed in the function to perform the specified task.

3) Is it necessary to use compatible data types in function definition and function call? Justify your answer with an example.

Ans: Yes, it is necessary to use compatible data types in function definition and function call because parameters pass in function call must have same data type of arguments declared in the function definition otherwise type mismatch error will be occur during compilation time.This is illustrated here through following example:

Example:

```
#include <stdio.h>
void fun (int x, int y)
}
X = 20;
Y = 10;
printf ("Values of x and y in fun (): %d %d", x, y);
}
void main ()
{
int x = 10, y = 20;
fun (x, y);
printf ("Values of x and y in main (): %d %d", x, y);
}
```

Output:

Values of x and y in fun (): 20 10
Values of x and y in main (): 10 20

4) Describe the advantages of using functions.

Ans: Functions provide us several advantages.

1. **<u>Reusability:</u>**

Functions provide reusability of code. It means that whenever we need to use the functionality provided by the function, we just call the functions. We do not need to write the same set of statements again and again.

2. **<u>Separation of tasks:</u>**

Functions allow us to separate the code of one task from the code of other tasks. If we have a problem in one function, then we do not need to check the whole program for removing the problem. We just need to focus at one single function.

3. Handing the complexity of the problem:

If we write the whole program as a single procedure, management of the program becomes difficult. Functions divide the program into smaller units and thus reduce the complexity of the problem.

4. Readability:

Dividing the program into multiple functions, improves the readability of the program.

5) What do you know about the return keyword?

Ans: Return is a keyword that is used to return a value to the calling function. Output of the function is called its **return value**. A function cannot return more than one value. If we try to get more than one value then compiler gives an error. There may be multiple return statement in a function but as soon as the first return statement is executed, the function call returns and further statements in the body of function are not executed.

Q4. Identify the errors in the following code segments. (K.B+U.B+A.B)

Ans:

Program	Error

a) void sum (int a, int b)
{
Return a + b;
}
	A function cannot return more than one value. e.g the following statement results in a compiler error.
	return a+b;

b) void message ();
{
printf ("Hope you are fine:)");
return 23;
}
	Statement terminator cannot be used at the end of function brackets. ( );

c) int max (int a; int b)
{
if (a > b)
return a;
return b;
}
	Error expected in parameterized function syntax (int a;int b) statement terminator cannot be used in brackets.

d) int product (int n1, int n2) return n1* n2;	A function cannot return more than one value. e.g the following statement results in a compiler error. Return n1*n2;

e) int totalDigits(int x)
{
 	
int count =0;
for(int i =x; i>=1, i=i/10)
count++;
return count
};
	• Syntax error expected in for loop structure for(int i=x;i>=1,i/10) comma cannot used at the place of statement terminator.
	• Statement terminator cannot be used at the end of body of function.

Q5. Write down output of the following code segments.

Ans:

Program	Output

a) int xyz (int n)
{
return n + n;
}
	20

int main ()
{
int p = xyx(5);
p = xyz(p);
printf("%d ",p);
}	

b) void abc (int a, int b, int c)
{
int sum = a + b + c;
}
int main()
{
int x = 4, y = 7, z = 23, sum1 = 0;
abc (x, y, z);
printf ("%d %d %d" x, y, z);
}
	4 7 23

c) int aa (int x)
{
int p = x / 10;
x++;
p = p + (p*x);
return p;
}
int main()
{
printf ("we got %d ", aa(aa(23)));
}
	We got 260

d) float f3(int n1, int n2)
{
n1 = n1 + n2;
n2 = n2 – n1;
return 0;
}
int main()
{
printf("%f\n", f3(3, 2));
printf("%f\n", f3(10, 6));
}
	0.000000
	0.000000

PROGRAMMING EXERCISES (A.B)

Exercise 1

Write a function int square(int x); to calculate the square of an integer x.

Solution:

```
#include<stdio.h>
        int square(int x)
        {
         return(x*x);
        }
        int main ( )
        {
                int x,n;
                printf("Input any number for square");
                scanf("%d",&x);
                n=square(x);
                printf("The square of %d is : %d",x,n);
                return 0;
         }
```

Exercise 2

Write a function int power(int x, inty); to calculate and return x^y.

Solution:

```
# include <stdio.h>
#include <math.h>
int power(int a, int b);
int main ( )
{
        int a,b,res;
        printf("Enter a : ");
        scanf ("%d",&a);
        printf("Enter b : ");
        scanf ("%d",&b);
   res=power(a,b);
   printf("Result: %d",res);
}
int power(int a,int b)
{
        int x;
        x=pow(a,b);
        return x;
 }
```

Exercise 3

Write a function to calculate factorial of a number.

Solution:

```
#include<stdio.h>
#include<math.h>
```

```
int main()
{
        printf("Enter a Number to Find Factorial: ");
        printf("\nFactorial of a Given Number is: %d", fact());
        return 0;
}
int fact()
{
        int i,fact=1,n;
        scanf("%d",&n);
        for(i=1; i<=n; i++)
    {
        fact=fact*i;
    }
        return fact;
 }
```

Exercise 4

Write a function which takes values for three angles of a triangle and prints whether the given values make a valid triangle or not. A valid triangle is the one, where the sum of three angles is equal to 180.

Solution:

```
#include <stdio.h>
int main( )
{
int side1, side2, side3;
printf("Enter the Lengths of Three Sides of a Triangle\n");
scanf("%d %d %d", &side1, &side2, &side3);
if((side1 + side2 > side3)&&(side2 + side3 > side1) &&(side3 + side1 > side2))
{
printf("It is a Valid Triangle\n");
}
else
{
printf("It is an invalid Triangle");
}
return 0;
}
```

Exercise 5

Write a function which takes the amount and the interest percentage and return the interest amount

Solution:

```
#include<stdio.h>
// function declaration
double calculateInterest(double p, double r);
// main function
int main()
{
```

```
                // declare variables
                 double p, r, interest;
                // take input from end-user
                printf("Enter principal amount and rate:");
                scanf("%lf %lf",&p,&r);
                 // calculate interest
                 interest = calculateInterest(p, r);
                 // display result
                 printf("Interest=%lf\n", interest);
                return 0;
}
// function to calculate interest value
double calculateInterest(double p, double r)
{
        return (p*r)/100;
 }
```

Exercise 6

Write a function which takes a number as input and displays its digits with spaces in between.

Solution:

```
#include <stdio.h>
#define MAX 100
// Function to print the digit of number N
void printDigit(int N)
                {
                 // To store the digit of the number N
                 int arr[MAX];
                 int i = 0;
                 int j, r;
                 // Till N becomes 0
                 while (N != 0) {
                 // Extract the last digit of N
                 r = N % 10;
                 // Put the digit in arr[]
                 arr[i] = r;
                 i++;
                 // Update N to N/10 to extract next last digit
                 N = N / 10;
                 }
                 // Print the digit of N by traversing arr[] reverse
                 for (j = i - 1; j > -1; j--)
                 {
                 printf("%d ", arr[j]);
                 }
```

```
}
int main()
{
int N = 3452897;
printDigit(N);
return 0;
}
```

Exercise 7

Write a function to print the table of a number.

```
#include<stdio.h>
int table(int n);
int main()
{
int n=0;
printf("Enter Number: ");
scanf("%d",&n);
table(n);
}
int table(int n)
{
int t=1;
printf("Table of  %d is:\n ",n);
for(int i=1; i<=10; i++)
{
t=n*i;
printf("\n%d x %d=%d",n,i,t);
}
return 0;
}
```

ANSWER KEY

5.1 FUNCTIONS

5.1.1 TYPES OF FUNCTIONS

5.1.2 ADVANTAGES OF FUNCTIONS

5.1.3 SIGNATURE OF A FUNCTION

5.1.4 DEFINING A FUNCTION

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C	A	B	B	B	A	C	A	A	D	D	C	A	A	A
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
B	C	B	A	A	A	A	C	C	C	C	D	B	A	A
31	32	33	34	35										
C	B	B	C	A										