CHAPTER 2

PROGRAMMING IN C

SHORT AND LONG QUESTIONS

Q1. Define computer programming.

Ans: Computer Programming:

Computer Programming is the process of writing a computer program in computer language to solve a particular problem.

Computer program controls the operation of a computer and it is developed in a computer language to perform a task. Computer languages are also known as programming languages.

Q2. Define Programming Language. Explain low level and high-level languages.

OR

Explain popular programming languages and their uses.

Ans: Programming Language:

A programming language is a language which is understood by computer. It is designed to give instructions to a computer to perform a specific task. It is used to write computer programs.

Classification of Programming Languages:

Programming languages can be classified into two categories that is low level language and high-level language.

i. **Low Level Languages:**

Low level language is machine-oriented language. To understand low level language, detailed knowledge of internal working of computer is

required. Low level languages include machine language and assembly language.

Classification of Low Level Languages:

- **Machine Language**

Programming language that is directly understood by computer hardware is known as machine language. Machine language is associated with architecture of computer. Therefore, programs written in machine languages for one computer will not work on another because of design differences. It consists of zeroes and ones. It is almost impossible for humans to use machine language because it entirely consists of numbers. Therefore, practically no programming is done in machine language instead, assembly language and high-level languages are used.

- **Assembly Language:**

Assembly language consists of symbolic codes or abbreviations known as mnemonics. It was developed to make computer programming easier than machine language. The abbreviations used in assembly language make it easier to learn and write programs compared to machine language. A program written in assembly language must be converted into machine language before it is executed by computer. A program known as assembler is used to translate assembly language into machine language.

Characteristics of Assembly Language:

Some important characteristics of Assembly language are

- Assembly language allows programmers to have access to all the special features of the computer they are using. Certain types of operations which are not possible in high-level languages are easily programmed using assembly language.

- Generally, a program written in assembly language will require less storage and less running time than one prepared in a high-level language.
- Assembly languages are still the best choice in some applications but their use is gradually declining.

ii. **High-level Languages (HLLs):**

High-level languages are English-oriented languages and they are commonly used for writing computer programs. These languages use English language word such as print, goto, if, end etc. Therefore, they are easy to learn and use.

Examples: Some examples of high-level languages are Visual Basic, C, Java and Pascal.

A program known as complier/interpreter is required to translate a high-level program into machine language. Coding and debugging of a high-level language program are much easier than a program written in a low level language.

Classification of High-level Languages:

High-Level languages can be classified into procedural, structured and object-oriented programming languages.

- **Procedural Languages:**

Procedural programming is based upon the concept of modular programming. In modular programming, programs are divided into smaller parts known as modules. Modular programs consist of one or more modules. A module is a group of statements that can be executed more than once in a program. Each module in a program performs a specific task.

Advantages of procedural languages:

It is easy to design, modify and debug a program in a procedural language since it provides better programming facilities.

Example of procedural languages:

Some examples of procedural languages are FORTRAN, Pascal, C and BASIC.

- **Structured Languages:**

Structured languages consist of three fundamental elements, which are sequence, selection and repetition.

Sequence:

It means writing program statements in a logical sequence. Each step in the sequence must logically progress to the next without producing any undesirable effects.

Selection:

It allows the selection of any number of statements based on the result of evaluation of a condition which may be true or false. Examples of statements that implement selection in programming are if, else-if, switch etc.

Repetition (loop)

It means executing one or more statements a number of times until a condition is satisfied. Repetition is implemented in programs using statements, such as for and while loops.

Some examples of structured languages are ALGOL, PL/1, Ada and Pascal.

- **Object-Oriented Programming Languages (oops):**

Object-orient programming (oops) refers to a programming method that is based on objects such as student, vehicle, building etc. Object-oriented programming language provides a set of rules for defining and managing

objects. An object can be considered a thing that can perform a set of activities.

For example, the object vehicle can be defined as an object that has number of wheels, number of doors, color, number of seats etc. The set of activities that can be performed on this object include Steer, Accelerate, Brake etc.

The most widely used object-oriented programming language are C++, C#, php and Java.

Q3. What is programming environment?

OR

Describe the concept of integrated development environment.

Ans: Programming Environment:

Programming environment is the set of process and programming tools used to develop computer programs. In the past, programmers used various standalone programming tools for developing computer programs. These included editors, complier, linker, debugger etc. Using separate programs provided a difficult and time-consuming environment for creating computer programs.

Today, programmers use Integrated Development Environment for developing computer programs. Integrated Development Environment is an application package that consists of editor, compiler, linker and debugger with a graphical user interface.

Q4. Describe different integrated development environment modules of C language.

OR

Define the following terms of C languages (Editor, Compiler, Linker, Loader and Debugger)

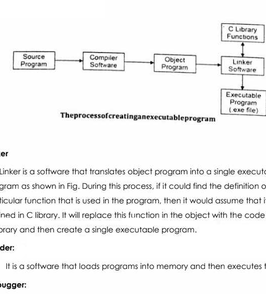
Ans: Programming Environment of C language:

A C language IDE consists of the following modules

- Text Editor
- Compiler
- Linker
- Loader
- Debugger

Text Editor:

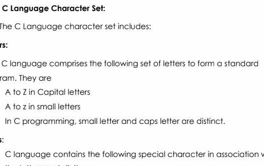
A text editor is a simple word processor that is used to create and edit source code of a program as shown in Fig. Files created by a text editor are plain text files. Most of the editors automatically highlight compile errors to simplify removing them.



Text editor used for creating and editing a program

Compiler

A compiler is a computer software that translated C language program (source program) into machine code (object program) that can be understood and executed by the computer as shown in Fig. It also detects syntax errors in the source program and gives hints to the programmer to correct them. A compiler will only create object program if all the syntax errors in a program have been corrected if they existed. A program written in C language has the extension .c (for example first c) and after compilation the object program extension will be .obj (for example first.obj)



Theprocessofcreatinganexecutableprogram

Linker

Linker is a software that translates object program into a single executable program as shown in Fig. During this process, if it could find the definition of a particular function that is used in the program, then it would assume that it is defined in C library. It will replace this function in the object with the code from C library and then create a single executable program.

Loader:

It is a software that loads programs into memory and then executes them.

Debugger:

It is a software that executes a program line by line, examines the values stored in variables and helps in finding and removing errors in programs.

Q5. List the uses of C language.

Ans: uses of C language:

C is a popular and widely used programming language for creating computer programs. A large variety of application programs and many modern operating system are written in C.

Q6. Explain C language character set.

Ans: C Language Character Set:

The C Language character set includes:

Letters:

C language comprises the following set of letters to form a standard program. They are

- A to Z in Capital letters
- A to z in small letters
- In C programming, small letter and caps letter are distinct.

Digits:

- C language contains the following special character in association with the letters and digits

Symbol	Meaning	Symbol	Meaning	Symbol	Meaning
~	Tilde	_	Underscore	]	Right bracket
!	Exclamation mark	+	Plus sign	:	Colon
#	Number sign	|	Vertical bar	"	Quotation mark
$	Dollar sign	\	Backslash	;	Semicolon
%	Percent sign	'	Apostrophe	<	Opening angle bracket
^	Caret	-	Minus sign	>	Closing angle bracket
&	Ampersand	=	Equal to sign	?	Question mark
*	Asterisk	{	Left brace	,	Comma
(	Left parenthesis	}	Right brace	.	Period
)	Right parenthesis	[	Left bracket	/	Slash

Q7. Define header files.

Ans: Header Files:

C language contains a number of standard functions in library file that performs various tasks in C programs. These tasks include all the input/output

operation and all the math operations. Library file contains header files and each header file contains a set of functions.

Q8. Write the uses of header files.

Ans: Uses of Header Files:

Some commonly used header files are stdio.h, conio.h and math.h

- Some commonly used functions that are included in stdio.h file and printf(), scanf() etc.
- Some of the functions that are included in conio.h file are clrscr(), getch() and getche().
- The math.h header file contains mathematical function such as sqrt(), pow(), log(), sin(), cos() and tan().

In order to use a library function, the programmer must include appropriate header file at the beginning of the program as shown in figure. When source program is translated into executable file, a copy of code of function is pasted in the source program by the linker from the header file before creating the executable file.

UseofheaderfileinaCprogram

Q9. What is the structure of a C program?

Ans: Structure of a C Program:

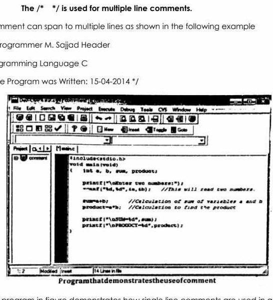
The formal according to which a program is written is called the structure of program. The following I the structure of C program.

Preprocessor directives
Global declarations
Main function
{
Local declaration
Statements } Body of main() function
}
User-defined function
Function1
Function 2
Function 3 } (option to user)

To introduce the structure of a C program, consider a very simple program that will print the message.

I love Pakistan:

The program may be written as shown in Fig.



Programtoprintamessageonthescreen

This program consists of three main parts which are preprocessor directive, the main() function and the body main().

Q10. Explain the types of comments in C language:

Ans: Types of Comments:

There are two types of comments.

i. Single line comment
ii. Multiple line comment

i. **Single Line Comment**

The // is used as single line comment. The syntax of single comment is // comment

An example is:

// Programmer: M. Sajjad Header

ii. **Multiple Line Comment:**

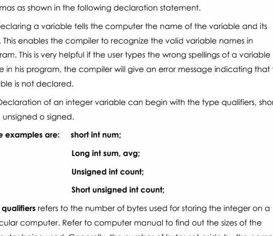
The /* */ is used for multiple line comments.

Comment can span to multiple lines as shown in the following example

/* Programmer M. Sajjad Header

Programming Language C

Date Program was Written: 15-04-2014 */



Programthatdemonstratestheuseofcomment

The program in figure demonstrates how single line comments are used in a program. In this program comments describe the purpose of statements.

Q11. Explain the different data types used in C programs.

Ans: Data Types Used in C Programs:

C provides three main data types for variables, that is, integer, floating-point and character variable.

- **Integer Data Type:**

Integer variable declaration statement has the form:

Type specifier Variable;

For example: int sum;

The declaration consists of the type name, int, followed by the name of the variable, sum. All the variables used in a C program must be declared. If there are more than one variable of the same type, separate the variable names with commas as shown in the following declaration statement.

Declaring a variable tells the computer the name of the variable and its type. This enables the compiler to recognize the valid variable names in program. This is very helpful if the user types the wrong spellings of a variable name in his program, the compiler will give an error message indicating that the variable is not declared.

Declaration of an integer variable can begin with the type qualifiers, short, long, unsigned o signed.

Some examples are: short int num;

Long int sum, avg;

Unsigned int count;

Short unsigned int count;

Type qualifiers refers to the number of bytes used for storing the integer on a particular computer. Refer to computer manual to find out the sizes of the computer being used. Generally, the number of bytes set aside by the compiler for the above type qualifiers are as given in table.

Integer data types used in C:

Variable Declaration	No. of Bytes	Range
--	2	-32,768 ~ +32,767
Long int	4	-2,147,483,648 ~ +2,147,483,647
Unsigned int	2	0 ~ 65,535

Here unsigned implies that the value of variable is greater than or equal to zero. The qualifier, signed is rarely used since by default, an int declaration assumes a signed number. Note that in the above examples, the word int may be omitted when it begins with long or unsigned type qualifiers in the declaration statements.

For example, you can write: **short num;**

- **Floating-point Data Type:**

Floating-point variables are used for storing floating-point numbers. Floating point numbers are stored in memory in two parts. The first part is the **mantissa** and the second part is the **exponent.**

The **mantissa** is the value of the number and the **exponent** is the power to which it is raised. Some examples of floating-point variable declaration statements are:

Float base, height;

Double float a, b, total;

Long double float size, x,y;

Usually the number of bytes set aside by the compiler for the above floating-point type qualifiers are as given in table.

Floating-point data types used in C:

Variable Declaration	No. of Bytes	Range
Float	4	$10^{-38} \sim 10^{38}$ (with 6 or 7 digits of precision)
Double float	8	$10^{-308} \sim 10^{308}$ (with 15 digits of precision)
Long double float	10	$10^{-4932} \sim 10^{4932}$ (with twice the precision of double)

Here, precision means the number of digits after the decimal point. The word float may be omitted when it is preceded by double or long double type qualifiers in the declaration statements.

There is another method of representing floating-point numbers known as exponential notation.

Example:

For example, the number 23,688 would be represented as 2.3688e4. Here 23688 is the value of the number (mantissa) and 4 is the exponent. The exponent can also be negative.

Example:

For example, the number 0.0005672 is represented as 5672 in exponential notation. Exponential notation is used to represent very large and very small numbers. In C, a floating point number of type float is stored in four bytes. Three

bytes for the mantissa and one byte for the exponent and provides 6 or 7 digits of precision.

- **Character Data Type:**

Character variables are used to store character constants. Examples of declaration of character variable are:

Char ch, letter, a;

A character variable can only store one character. The compiler sets aside one byte of memory for storing a character in a character variable.

Q12. Define constant qualifier.

Ans: The Constant Qualifier:

The constant qualifier is used with a data type declaration to specify a constant value.

Examples: Some examples of const qualifiers are

Const int COUNT= 5;

Const float PI=3.14;

The same thing can also be achieved using #define preprocessor as given below

#define COUNT 5

#define PI 3.14

In most of the situations there is no difference between using define or const qualifier. The only difference is defined is a preprocessor directive that simply replaces a symbol with a constant value throughout the program. The const qualifier declares a variable type and assign it a constant value. It has some advantage at advance level of programming in C language.

Q13. Explain the main () Function.

Ans: The main () Function:

A C program consists of one or more functions. C uses functions as the building block of its programs. A function performs a single well-defined task. Every C program must have the function main () which is the first section to be executed when the program runs.

The word void before the function main () means that this function does not return a value and the second void inside the brackets means it does not have any argument.

A program may also contain user-defined functions. User-defined functions are written by programmers and they are included in the library file. These functions may be written before or after the main () function and are used inside the main () function. When we run a program, the main () function is always executed first no matter where it appears in the program.

The Body of main () Function:

The body of the function main () is surrounded by braces (curly brackets { and }). The left brace indicates the start of the body of the function and the matching right brace indicates the end of the body of the function.

The body of the function in the program (see Figure) consists of a single statement printf() which ends with a semicolon(;) printf() is the standard output function. The text to be printed is enclosed in double quotes. A statement in C is terminated with a semicolon. Therefore, a semicolon is used at the end of **printf()** statement. This program will print the message **I Love Pakistan** on the screen.

Q14. Differentiate between Global and Local Variables.

Ans: Global and Local Variables:

Variables are used in C programs to store values. All the variables used in a program must be declared. There are two types of variables, global variables and local variables.

In the program structure, global variables are declared before the main() function and local variables are declared within it.

Global variables are those variables which can be used in any function in the program whereas local variables are used only within a function in which they are declared.

KEY POINTS

- **Syntax** refers to the rules of a programming language according to which statements of a program are to be written.
- **Semantic** gives meaning to statements of a programming language.
- **Machine language** is a programming language that is directly understood by computer hardware. It consists of zeroes and ones.
- **Assembly language** consists of symbolic codes or abbreviations. It was developed to make computer programming easier than machine language. A program called assembler is required to convert assembly language to machine language for execution by the computer
- **High-level language** is an English-oriented language. It is commonly used for writing computer programs. A program called compiler/interpreter is required to translate high-level language into machine language for execution by the computer.
- **Compiler** is a computer program that translates a program written in a high-level language into machine language equivalent program. It converts the entire program into machine language before execution by the computer.

- **Interpreter** is a computer program that translates a program written in a high-level language into machine language. It translates one instruction at a time and executes it immediately before translating the next instruction.
- **Programming environment** is the set of processes and programming tools used to develop computer programs.
- **Integrated Development Environment (IDE)** is a computer software that is used to create, compile and run programs. It brings all the processes and tools required for program development into one place.
- **Text editor** is a simple word-processor that is used to create and edit source code of a program.
- **Linker** is a software that takes one or more object files, generated by a compiler and combines them into a single executable program.
- **Loader** is a software that loads programs into memory and then executes them.
- **Debugger** is a software that executes a program line by line, examine the values stored in variables and helps in finding and removing errors in programs.
- **Header file** is a file that contains predefined functions of C language. Including a header file in a C source produces the same result as copying the header file into the source file.
- **Reserved words** are those words that are part of a programming language and have special purpose in computer programs.
- **Preprocessor directives** are instructions for the C compiler at the beginning of program.
- **Comments** are notes that are included in programs when a fact is necessary to be brought to the attention of program's reader.
- **Constant** is a quantity whose value does not change during program execution.

- **Variable** is a symbolic name that represents a value that can change during execution of a program.
- **Type casting** is a way to convert a variable from one data type to another data type.

EXERCISE

Q1. Select the best answer for the following MCQs.

i. Which of the following is not a valid variable name in programming?
A. 2variable
B. variable2
C. _byVar
D. 2ndVariable

ii. Which language is directly understood by the computer?
A. Assembly language
B. Machine language
C. High-level language
D. Pseudocode

iii. What was C language developed?
A. 1970
B. 1969
C. 1972
D. 1980

iv. Who is developed Java language?
A. Dennis Ritchie
B. Bjarne Stroustrup
C. James Gosling
D. Alan

v. What is the other word used for Reserved Words?
A. Keywords
B. Identifiers
C. Special programming words
D. Statements

vi. How many bytes are set aside by the compiler for a variable of type int?
A. 1
B. 2
C. 4
D. 8

vii. How many bytes are set aside by the compiler for a variable of type float?
A. 2
B. 4
C. 8
D. 16

viii. What is the type of hardware that can be stored in a variable of type double float?
A. 32 bit - 64 bit
B. 64 bit
C. 127 - 255
D. 1075 - 2047

ix. Which program translates high-level language into machine language?
A. Compiler
B. Linker
C. Loader
D. Debugger

x. Which software helps in finding and correcting errors in programs?
A. Compiler
B. Text Editor
C. Loader
D. Debugger

Answer Key

i	ii	iii	iv	v	vi	vii	viii	ix	x
B	B	A	C	A	C	B	B	A	D

SHORT QUESTIONS

Q2. Give short answers to the following questions:

i. Define computer program.

Ans. Computer Program

A computer program is a set of instructions written in a programming language to solve a specific problem and achieve specific results. Programs can be simple, like calculating the sum of two numbers, or complex, like running an operating system.

For example, an assignment statement consists of a variable and an expression separated by the assignment operator (=). The value of the expression is stored in the variable on the left side of the assignment operator.

Variable expression

Syntax:

Syntax rules are important to follow while writing a program. Each programming language has its own syntax rules that must be followed for the program to run correctly.

For example, in the statement int number = 5;, the syntax rules include the use of a semicolon (;) at the end of the statement.

Semantics:

Semantics refers to the meaning of the statements in the program. If the syntax rules are correct, but the statement is not what you intended, then the result is incorrect.

ii. Differentiate between syntax and semantic.

Ans. Syntax and Semantic

Syntax refers to the rules of a programming language according to which statements of a program are to be written. It describes the way in which instructions are written in a program. Semantics refers to the meaning of the statements in the program.

For example, an assignment statement consists of a variable and an expression separated by the assignment operator (=). The value of the expression is stored in the variable on the left side of the assignment operator.

Variable expression

Semantics:

Semantics refers to the meaning of the statements in the program. If the syntax rules are correct, but the statement is not what you intended, then the result is incorrect.

iii. What is the characteristics of HLL.

Ans. Characteristics of High level languages

- High-level languages are easy to understand and maintain.
- These languages are machine-independent, meaning they can be executed on different types of computers.
- High-level languages have built-in error-checking and debugging features, which help to identify and correct errors in the program.
- They are designed to be portable, meaning they can be transferred from one computer system to another with minimal changes.
- They provide a higher level of abstraction, which means they are further removed from the machine's hardware and instruction set.

Features of Assembly language

- Assembly language is a low-level language and is easy to understand than the machine language.
- The programs written in this language are not portable and the programs are machine-dependent.
- The programs developed in assembly language are not flexible and hard to be expanded.

iv. What four characteristics of HLL.

Ans. Characteristics of High level languages

- High-level languages are easy to understand and maintain.
- These languages are machine-independent, meaning they can be executed on different types of computers.
- High-level languages have built-in error-checking and debugging features, which help to identify and correct errors in the program.
- They are designed to be portable, meaning they can be transferred from one computer system to another with minimal changes.
- They provide a higher level of abstraction, which means they are further removed from the machine's hardware and instruction set.

Features of High level language (HLL)

- High-level languages are easily understandable.
- The programs that are developed in high-level languages are portable.
- The high-level language debugging of the code is easy and the program written is not dependent.

Features of Assembly language

- Assembly language is a low-level language and is easy to understand than the machine language.
- The programs written in this language are not portable and the programs are machine-dependent.
- The programs developed in assembly language are not flexible and hard to be expanded.

v. Differentiate between control and variable.

Ans. Controls and Variables

Controls and variables are used in experimental research to control confounding variables. Variables are the factors that researchers measure, manipulate, or control.

- **Control**

A control is a variable that we keep constant.

- **Variable**

A variable is a variable name that represents a value that can change during execution of a program. It is called a variable because the value it holds can be changed or varied.

Two key terms, character vs string

- **Numeric Constants**

Numeric constants are a set of any digits, integer and floating point numbers.

Integer constant represents value that can be stored in the number of bytes as per the maximum size of integer (constant like 7 or 10).

- **Floating point constants** are used to represent values that can be stored in the number of bytes as per the maximum size of float (10.23 or 0.0 or 1.0).

- **Character Constant**

Character Constant is one of the symbols A-Z character set (A character digit 0-9, space case letters a-z, Special symbols & (,), [,], {, }, <, >, +, -, *, /, =, !, ?, ', ", ., ,, ;, :, $, %, ^, ~, #, |, \, @, _) enclosed within single quotation marks (e.g. 'a', 'A', '+', '5', '$', '@', '\n', ' ').

- **String Constant**

String constant contains a string of characters within double quotes (e.g. "Hello World", "12", etc.).

- **Variable**

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 an identifier, and a value that is stored in memory. The value of a variable can be changed as the program runs. A variable is like a box that holds the data that a program needs to work with.

Types of Variables

Variables are of two types, numeric and character.

- **Numeric Variables**

Numeric variables are used to represent numeric values in computer programs. They represent integer and floating point values. Some examples of numeric variables are sum, length, radius, mass, etc.

- **Character variables**

Character variables represent character values in computer programs. They represent single characters, such as letters, numbers, or symbols. Some examples of character variables are ch, grade, letter, etc.

When a variable is used in a computer program, the computer allocates a memory location to store the value of the variable. The value of the variable can be changed by the program as it runs. The name of the variable and the value that it stores are separate. A name is a label that is used to refer to the variable. The value is the data that is stored in the memory location that is associated with the variable.

Q3. Which of the following are valid C variables? Give the reason if not a valid variable. (a) int, (b) 1st_value, (c) float, (d) char, (e) int, (f) 1st_value

Ans. Valid

- (a) int
- (c) float
- (d) char
- (e) int

Not valid

- (b) 1st_value: A variable name cannot start with a number
- (f) 1st_value: A variable name cannot start with a number

Q4. Why comments are used in programs?

Ans. Purpose of Comments

It is a good programming practice to add comments in program to make it easy for others to understand it. Comments in the source code are ignored by the compiler.

Comments are added in programs when a line or more lines need to be brought to the attention of a program's reader. Generally, programmers write comments to explain the purpose of a piece of code. Comments may help before the program is revisited, to make it possible the program code retains something to be read without slow the discovery of the program's purpose.

Q5. What is the purpose of header files in C language?

Ans. Purpose of Header file in C Language

Header files contain definitions of functions and variables which many programs would need to use in the C or C++ programming language. Header files include the function file at the beginning of the program by using the preprocessor #include statement. Header files provide access to the function's variables and methods. Whenever header files are included, they are saved with the .h extension.

EXTENSIVE QUESTIONS

Q1. Exercise the following High Level languages

a) ALGOL
b) Assembly Basic
c) C#
d) Java

Ans. Exercise High Level languages (HLLs)

a) ALGOL

A language was developed around 1970 by Tarvi Pohlonk at Yale University. It is a procedural programming language and is well suited for writing articles. The Algol language has had a significant impact on many languages that came after it, including BCPL, C, Pascal, and Ada. Algol is also well-known for its use of block structure.

b) Assembly Basic

Assembly Basic is a set of programming language and tools used to create software programs for computers based on Intel's processors.

Reverse and revise statements are also used for use by the other programs. Assembly Basic is a low-level language, which means it is very close to the instructions that a computer's processor can execute. It is considered an advanced version of Assembly Language 2 and 16 bit tested.

Assembly Basic is a low-level code assembly name since the compiler checks to distinguish between them. It is also the code is also not portable. Using assembly tool kit to combine some might cause accidental errors, which can be difficult to find and debug.

Q2 enhanced version of Q 1

Ans. Purpose of Header file in C Language

Header files contain definitions of functions and variables which many programs would need to use in the C or C++ programming language. Header files include the function file at the beginning of the program by using the preprocessor #include statement. Header files provide access to the function's variables and methods. Whenever header files are included, they are saved with the .h extension. Header files are also known as header or include files.

Some commonly used header files are stdio.h, conio.h, and math.h.

c) C#

C# language was developed around 2000 by Scott Wiltamuth and Anders Hejlsberg. It is intended to be a simple, modern, general-purpose, object-oriented programming language.

d) Java

James Gosling initiated the Java language project in June 1991 for use in one of his various set-top box projects. The language was originally called Oak after an oak tree that stood outside Gosling's office. Later the project went by the name Green and ended up later renamed as Java, from a list of random words. Gosling's goals were to implement a virtual machine and a language that had a familiar C-like notation.

Q3. Write three differences between Assembly language and HLLs.

Ans. Difference between High level language and assembly language

Features of High level language (HLLs)

- High-level languages are easily understandable.
- The programs that are developed in high-level languages are portable.
- The high-level language debugging of the code is easy and the program written is not dependent.

Features of Assembly language

- Assembly language is a low-level language and is easy to understand than the machine language.
- The programs written in this language are not portable and the programs are machine-dependent.
- The programs developed in assembly language are not flexible and hard to be expanded.

Q4. Write four characteristics of HLLs.

Ans. Characteristics of High level languages

- High-level languages are easy to understand and maintain.
- These languages are machine-independent, meaning they can be executed on different types of computers.
- High-level languages have built-in error-checking and debugging features, which help to identify and correct errors in the program.
- They are designed to be portable, meaning they can be transferred from one computer system to another with minimal changes.
- They provide a higher level of abstraction, which means they are further removed from the machine's hardware and instruction set.

Features of High level language (HLLs)

- High-level languages are easily understandable.
- The programs that are developed in high-level languages are portable.
- The high-level language debugging of the code is easy and the program written is not dependent.

Features of Assembly language

- Assembly language is a low-level language and is easy to understand than the machine language.
- The programs written in this language are not portable and the programs are machine-dependent.
- The programs developed in assembly language are not flexible and hard to be expanded.

Q5. What are the criteria for selecting suitable names in C language?

Ans. How to Name Identifiers in C Language

A variable is a symbolic name that represents a value that can change during execution of a program. It is called a variable because the value it holds can be changed or varied.

The following are the rules for specifying suitable names in C language:

- A variable begins with a letter or underscore (_) and can be followed by letters, digits, or underscores.
- Variable names are used to improve readability of the variable name.

- The keywords, int, char, etc.
- There is no restriction on the length of a variable name. However, only the first 31 characters are significant. The rest are ignored. Therefore, two variables that have the same 31 characters at the beginning are treated as the same variables.
- Upper case letters and lower case letters are allowed in naming variables, so upper case letter is considered different from its corresponding lower case letter. For example, the variable AVG is different from its string, avg.
- Variable names cannot be the same as any of the C keywords, such as int, char, float, etc.
- Spaces or not allowed and hence must be no embedded blank in the name of a variable. For example, max no is not a valid variable name.

Q6. What is a preprocessor directive? Explain #include preprocessor directive in detail.

Ans. Preprocessor Directives

Preprocessor directives are directives for the C compiler. Every C language program is processed before it enters the phase of the beginning of the program. Before functioning of C language program the machine language file. The preprocessor directives are also known as Preprocessor directives. These directives are used to arrange sign at

The most commonly used preprocessor directives are #include and #define.

- **#include Preprocessor Directive**

Prior the following syntax:

#include<header file name>

When this preprocessor is carried out by the C compiler, it will search for the header file in the standard library and then it will include generated files in it. symbol and copies them to the source.

For example, if the **#include** header file is written within the text then and the compiler will search for the header file in the standard library. Then the compiler will continue to copy the **#include** header file to the program. The **#include** header file stands for standard input output header. It includes the data about **printf()** and **scanf()** functions. Therefore, the **#include** directive is used to include the data and the necessary library functions that are included in it.