3) int my variable name;
4) int void = 5;
5) int nAngle;
6) int 3some;
7) int meters_of_pipe;
8) int length, width=5;

Q.6 Write a C++ program to get six subjects marks of a student and then calculate its total, average and percentage and display them on the screen.

Q.7 Write a C++ program to find out maximum value out of three integers using conditional operator.

Q.8 Write a C++ program to find out the roots of a quadratic equation.

Q.9 Write a C++ program to find out the area of a rectangle and display the result on the screen.

Q.10 Write a C++ program to get the age of a student from the user at run time and display it on the screen.

UNIT 4

CONTROL STRUCTURE

After the completion of Unit-4, Students will be able to:

- Explain the use of the following decision statements:
  - **if** statement
  - **if-else** statement
  - **switch** statement
- Know the concept of nested **if** statement
- Use **break** statement and **exit** function
- Explain the use of the following looping structure:
  - **for** loop
  - **while** loop
  - **do-while** loop
- Use **continue** statement
- Know the concept of nested loop

INTRODUCTION

C++ provides **control flow statements,** also called *flow control statements*, which allows the programmer to control the sequence of execution of statements of a program by the processor. The most commonly used control flow statements are decision, repetitions and compound statements.

4.1 Decision statements

Decision statement is a statement that causes the program to change the path of execution, sequence of execution, based on the value of an expression. It is also called conditional statement. Most commonly used decision support statements are: if, if-else and nested if-else statements. The switch also provides a mechanism for doing conditional branching.

4.1.1 Use of decision statements

In a C++ program, a programmer sometimes needs to execute one statement (or a set of statements) if a certain condition is true and another statement (or set of statements) if it is false. In a C++ program, this is only possible with the use of decision statements.
The following are different types of decision statements.

a. if statement

The most basic kind of conditional statement used in C++ is **if** statement. General syntax of an *if statement* is given below:

Single statement if structure

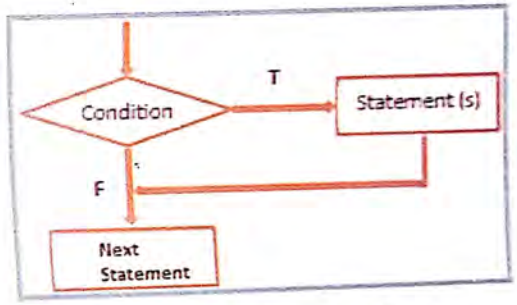
```
if (condition)
Statement;
```


Multiple statements if structure

```
if (condition)
{
Statement 1;
Statement 2;
.
.
Statement n;
}
```

If the condition is true, the statement after **if** will be executed and if the condition is false, the statement will be skipped. If we want to execute multiple statements with one **if** statement then the statements should be enclosed in a pair of curly braces { }.

Flowchart 4.1 shows working of **if statement.**



Flowchart 4.1: Working of If Structure

Consider the following program to demonstrate the use of **if** statement.

```
#include <iostream.h>
#include<conio.h>
```

```
int main()
{
int X;
cout << "Enter a number: ";
cin >> X;
if (X %2==0)
cout << X << " is even number";
getch();
return 0;
}
```

Output of the Program

```
Enter a number:
12
12 is even number
```

b. if-else statement

In **if** construct, the statements in the body of **if** are executed only if the condition is true and nothing is executed if it is false. Sometimes the programmers need to execute some other part of a program even if the condition in **if** statement is false. For this purpose, *if-else statement* is used.

The general syntax of this statement is given below.

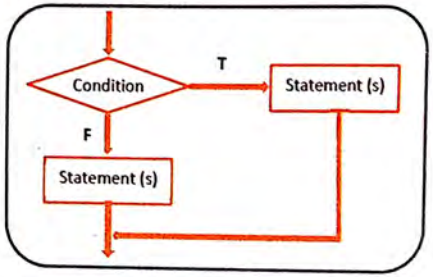
Single statement if-else structure

```
if (condition)
Statement;
else
Statement;
```

Multiple statements if-else structure

```
if (condition)
{
Statement 1;
Statement 2;
.
.
Statement n;
}
else
{
Statement 1;
Statement 2;
.
.
Statement n;
}
```

If the condition is true, the body of **if** is executed. If the condition is false, the body of *else* is executed. Flowchart 4.2 shows working of **if –else Structure**.



Flowchart 4.2: Working of If-else Structure

Consider the following program to demonstrate the use of *if-else statement*.

```
//if-else (even/odd) program
#include <iostream.h>
#include<conio.h>
int main()
{
int X;
cout << "Enter a number: ";
cin >> X;
if (X %2==0)
cout << X << "is even number";
else
cout << X << " is odd number";
getch();
return 0;
}
```

Output of the program

```
Enter a number:
9
9 is odd number
```


In order to execute multiple statements, a block should be used. Consider the following program:

```
//if-else (multiple statements) program
#include <iostream.h>
#include<conio.h>
int main()
{
int X;
cout << "Enter a number: ";
cin >> X;
if (X%2==0)
{
cout << "You have entered " << X << endl;
cout << X << "is an even number" << endl;
}
else
{
cout << "You have entered " << X << endl;
cout << X << " is an odd number" << endl;
}
getch();
return 0;
}
```

Output of the program

```
Enter a number:
08
You have entered 08
08 is an even number
```

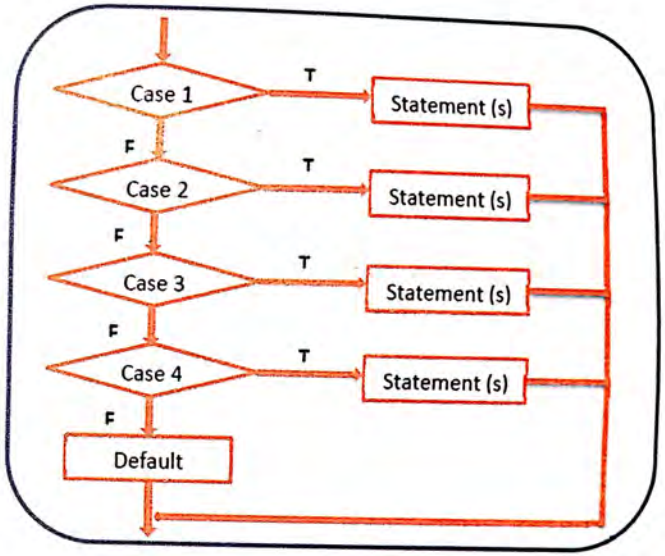
c. Switch-default statement

Switch statement is a better alternative of the nested if-else structure. The nested if-else structure is sometimes more difficult to understand to find that which if clause is associated with which else clause. The objective of switch statement is to check several possible constant values for an expression and then execute those which match the particular label or case constant value. Its general form is as follows:

```
switch (expression)
{
case constant1:
group of statements 1;
break;
case constant2:
group of statements 2;
break;
.
.
.
default:
default group of statements
}
```

switch evaluates the expression and if the value of the expression equal to constant1, it executes group of statements 1 until it encounters the **break** statement. When it reaches the **break** statement the program jumps to the end of the switch. If the value of the expression is not equal to constant1, it is matched with constant2. If it is equal to this constant2, this group of statements 2 is executed until the **break** keyword is reached, and then a jump will occur to the end of the switch. Finally, if the value of expression does not match with any of the constants then the program executes the statements after the

default keyword. The **default** clause is optional. Flowchart 4.3 shows working of **switch** **Structure**.



Flowchart 4.3: Working of switch-default Structure

Consider the following two code segments which have the same behavior.

switch example	if-else equivalent

// Switch program
#include <iostream.h>
#include<conio.h>
int main()
{
int x;
cout<<"Enter value for x";
cin>>x;
switch (x)
{
case 1:
cout << "x is 1";
break;
case 2:
cout << "x is 2";
break;
default:
cout << "value of x is neither 1 nor 2";
}
getch();
return 0;
}
	// if-else alternative of the switch program
	#include <iostream.h>
	#include<conio.h>
	int main()
	{
	int x;
	cout<<"Enter value for x"<<end;
	cin>>x;
	if (x == 1)
	{
	cout << "x is 1";
	}else
	if (x == 2)
	{
	cout << "x is 2";
	}else
	{
	cout << " value of x is neither 1 nor 2";
	}
	getch();
	return 0;
	}

In the above program, if we did not include a break statement after the first group for **case** 1, the program will not automatically jump to the end of the switch and it would continue executing the rest of statements until it reaches either a break instruction or the end of the switch block.

switch statement can only be used to compare the value of the expression with the constants written with each **case** clause. Therefore, variables cannot be used with constants such as **case n** or ranges such as **case (1..3):** because they are not valid C++ constants. These constants be either integers or character type.

4.1.2 Nested if statement

It is possible to chain if-else statements together. When one if-else statement is used inside the body of another if-else structure then it forms a nested if ladder.

The following program demonstrates the use of nested if structure.

```
#include <iostream.h>
#include<conio.h>
int main()
{
int X;
cout << "Enter a number: ";
cin >> X;
if (X > 100)
cout << X << "is greater than 100" << endl;
else if (X < 50)
cout << X << "is less than 50" << endl;
else
cout << X << " is between 50 and 100" << endl;
getch();
return 0;
}
```

Output of the program

```
Enter a number:
96
96 is between 50 and 100
```

Consider another program demonstrating nested if-else structure by displaying whether an entered integer is multiple of 2 or 3 or is some other number.

```
#include <iostream.h>
#include<conio.h>
int main()
{
int n;
cout<<"Enter a number: ";
cin>>n;
if (n%2== 0)
cout << n<< " is multiple of 2";
else
if (n %3== 0)
cout << n<< " is multiple of 3";
else
cout << n<< " is neither multiple of 2 nor 3";
getch();
return 0;
}
```

Output of the program

```
Enter a number:
55
55 is neither multiple of 2 nor 3
```


Consider another example demonstrating nested if-else structure to display the grades of students. This program takes percentage marks as input and then using relational and logical operators find its corresponding grade.

```
#include <iostream.h>
#include<conio.h>
int main()
{
float per;
cout<<"Enter percentage marks: "<<endl;
cin>>per;
if (per >= 80)
{
cout << "Your grade is A+"<<endl;
cout<< "Excellent";
} else
{
if (per < 80 && per>=70)
{
cout << "Your grade is A"<<endl;
cout<< "Very good";
} else
{
if(per<70 && per>=60)
{
cout << "Your grade is B"<<endl;
cout<< "Good";
}else
cout<<"You are failed, try again";
```

```
}
}
getch();
return 0;
}
```

Output of the program

```
Enter percentage marks:
65.09
Your grade is B
Good
```

4.1.3 break statement and exit() function

Sometimes a programmer needs to make use of such statements in a program that are responsible for the jump from one statement of the program to another statement. The most commonly used jump statements of C++ language are: **goto, break,** and **continue.** Similarly, the execution of a running program is also sometime needed to be halt and to jump out of a program. A function used to accomplish this is **exit()**. The following sections discuss the use of **break** statement and **exit ()** function with the help of examples.

a. break statement

break statement is used to make jump from one part of a program to another. In C++, *break* statement is used in two situations: one in *switch-default* structure to properly end a *case* clause and another in loops structure to leave a loop. The use of *break* in *switch-default* structure has already been discussed in Section 4.1.3. Its use in loops will be discussed in Section 4.2 with the help of examples.

b. exit () function

This function tells the program to quit running immediately. This function is responsible for the immediate termination of a C++ program. It is defined in the *stdlib* header file. The *exit* function takes an integer, usually 0, as a parameter that is returned to the operating system as an exit code and tells it that the program has terminated normally.

Consider the following example for demonstrating the use of **exit** () function.

```
#include<conio.h>
#include <stdlib.h>
#include <iostream.h>
int main()
{
cout << "Statement before exit() function";
exit(0);
//terminate and return 0 to the operating system
// The following statements never execute
cout << "Statement after exit() function";
getch();
return 0;
}
```

Output of the program

```
Statement before exit() function
```

4.2 LOOPS

Loops are used to repeat a statement or group of statements a certain number of times until the condition is true. These are also called repetitive or iterative statements.

4.2.1 Types of loops

There are three types of loops.

- for
- while
- do-while

a. for loop

The most commonly used looping structure in C++ is **for** loop. **for** loop is ideal in situation when we know in advance the exact number of iteration. Its general syntax is:

Single statement for-loop

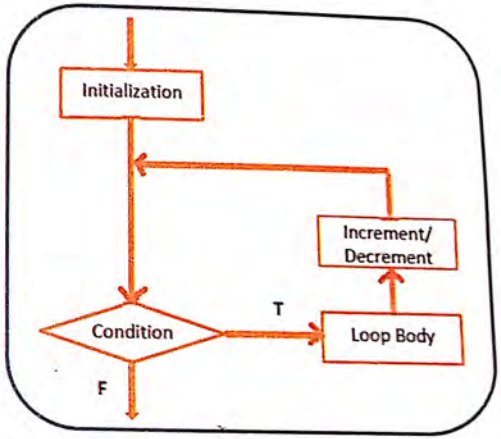
```
for (initialization; condition; increment/decrement)
statement;
```

Multiple statements for-loop

```
for (initialization; condition; increment/decrement)
{
Statement 1;
Statement 2;
.
.
Statement n;
}
```

In the above two general forms of *for* loop, the first one has only one statement in its body and has no need for braces { } while in the second form multiple statements are used so they are enclosed in the pair of braces { }.

The main function of **for** loop is to repeat statement(s) while condition remains true. It works in the following way shown in Flowchart 4.4.



Flowchart 4.4: Working of for Loop

1. **Initialization**: Generally it is an initial value setting for a counter variable. This is executed only once.
2. **Condition**: If it is true the loop continues, otherwise the loop ends and statement is skipped (not executed).
3. **Loop Body**: It can either be a single statement or a block enclosed in braces { }, for multiple statements.
4. Finally, the loop counter is incremented/decremented and control goes back to step2.

Consider the following program to display the string PAKISTAN five times on the screen.

```
#include <iostream.h>
#include<conio.h>
```

```
int main()
{
for (int i=1; i<6; i++)
cout << "PAKISTAN"<<endl;
getch();
return 0;
}
```

Output of the program

PAKISTAN
PAKISTAN
PAKISTAN
PAKISTAN
PAKISTAN

Here is an example of countdown using **for** loop:

```
// countdown using a for loop
#include <iostream.h>
#include<conio.h>
int main()
{
for (int n=10; n>0; n--)
cout << n << ", ";
getch();
return 0;
}
```


Output of the program

10, 9, 8, 7, 6, 5, 4, 3, 2, 1

The *initialization* and *increment/decrement* fields are optional. They can remain empty, but in all cases the semicolon signs between them must be written. For example, we could write: for (;n<10;) if we want to specify no initialization and no increase.

```
int n=0;
for ( ; n < 10; )
{
cout << n << ",";
n++;
}
```

Output of the code segment

0,1,2,3,4,5,6,7,8,9

Similarly, *for* loop can also be used as: for (;n<10;n++) if we want to include an increment/decrement expression but no initialization.

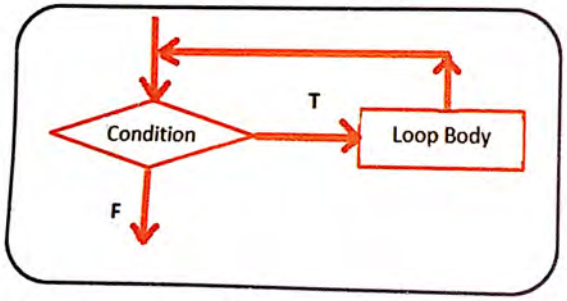
b. while loop

while loop is another structure that is used in computer programs to repeat a statement or a group of statements until the condition is true.

```
while (condition)
{
Statement 1;
Statement 2;
```

Statement n;
}

Flowchart 4.5 shows working of **while loop.**



Flowchart 4.5: Working of while Loop

Consider the following program to display the string PAKISTAN five times on the screen using **while** loop.

```
// displaying PAKISTAN 5 times using while loop
#include <iostream.h>
#include<conio.h>
int main()
{
int i=1;
while (i<6)
{
cout << "PAKISTAN"<<endl;
++i;
}
```



```
getch();
return 0;
}
```

Output of the program

PAKISTAN
PAKISTAN
PAKISTAN
PAKISTAN
PAKISTAN

Like **for** loop, the **while** loop has also three parts: *initialization, condition and increment/decrement.* The difference is that here the initialization is done before the **while** clause and the increment/decrement is done within the body of the **while loop.**

The second example of countdown using while-loop is given below.

```
// count down program using while loop
#include <iostream.h>
#include <conio.h>
int main()
{
int n;
cout << "Enter the starting number: ";
cin >> n;
while (n>0)
{
cout << n << ", ";
--n;
```

```
}
getch();
return 0;
}
```

Output of the program

Enter the starting number:
8
8, 7, 6, 5, 4, 3, 2, 1

Here, when the program starts the user enters a number to start the countdown. In this case, the number is 8 and thus the condition n>0 is true, therefore, the block within the braces { } is executed as long as n>0 and the output 8,7,6,5,4,3,2,1 is displayed on the screen.

Consider another program that prints numbers from 1 to 50 in such a way that there should be 10 digits per line.

```
#include <iostream.h>
#include <conio.h>
int main()
{
 int n = 1;
 while (n <= 50)
 {
  cout << n << " ";
  if (n % 10 == 0)
  cout << endl;
  n++;
```



```
}
getch();
return 0;
}
```

Output of the program

1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50

Consider another program using *while* loop to get a sentence from the user and count the number of characters in this sentence.

```
#include <iostream.h>
#include <conio.h>
int main()
{
char ch;
int NumberOfCharacters=0;
ch= getche();
while (ch!= '.')
{
++NumberOfCharacters;
ch= getche();
}
cout<< "The number of characters in the sentence are: "<<NumberOfCharacters;
getch();
```

```
return 0;
}
```

Output of the program

Welcome to the class.
The number of characters in the sentence are: 20

The main difference of the *while* loop and *for* is that in *while* loop it is not necessary to know the exact number of iterations of the body of the loop in advance as in *for loop*.

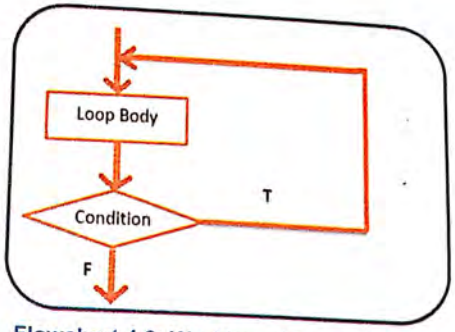
c. do- while loop

Sometime a programmer want to execute the loop at least once for some problems. To do this, C++ offers the facility of **do while** loop. The general format of a **do while** loop is:

```
do
{
Statement 1;
Statement 2;
.
.
Statement n;
}
while (condition);
```

The working of *do while* loop is the same as the *while* loop, except that the condition in *do while* loop is checked after the execution of the body of the loop. The execution of *do while* loop, guaranties the execution of the statements in its body, at least once, even if the condition is false.

Flowchart 4.6 shows working of **while loop.**



Flowchart 4.6: Working of do-while Loop

Consider the following program that prints the string PAKISTAN once although the condition is false in the start.

```
// displaying PAKISTAN at least once using do while loop
#include <iostream.h>
#include<conio.h>
int main()
{
int i=10;
{
cout << "PAKISTAN"<<endl;
++i;
} while (i<6);
getch();
return 0;
}
```

Output of the program

PAKISTAN

In the above program, the loop control variable 'i' is initially set to 10 and in the condition it is restricted to be less than 6 (which is violated in the start) but still it is executed once and have displayed the message PAKISTAN.

Consider another program that echoes any number you enter from the keyboard to the screen until you enter 0 using do-while loop.

```
#include <conio.h>
#include <iostream.h>
int main()
{
int x;
do {
cout << "Enter number (0 is for ending): ";
cin >> x;
cout << "You entered: " << x << "\n";
} while (x != 0);
getch();
return 0;
}
```

Output of the program

```
Enter number (0 is for ending): 12
You entered: 12
Enter number (0 is for ending): 16
You entered: 16
Enter number (0 is for ending): 0
You entered: 0
```


4.2.2 continue statement

The continue statement provides a convenient way to jump to the start of a next iteration by passing the remaining part of the loop for an iteration. For example, we are going to skip the number 5 in our countdown:

```
// use of continue statement in loop
#include <iostream.h>
#include <conio.h>
int main()
{
for (int n=10; n>0; n--)
{
if (n==5) continue;
cout << n << ", ";
}
getch();
return 0;
}
```

Output of the program

10, 9, 8, 7, 6, 4, 3, 2, 1

4.2.3 Nested loops

The use of a loop inside the body of another loop is called **nested** loop. The enclosing loop is called *outer* loop and the enclosed loop is called *inner* loop. The following example demonstrates the concept of nested loop.

```
// nested while loop program
#include <iostream.h>
```

```
#include <conio.h>
int main()
{
// Outer Loop between 1 and 5
int O=1;
while (O<=5)
{
int I = 1;
while (I <= O)
{
cout << I++;
}// end of inner loop
// print a newline at the end of each row
cout << endl;
O++;
}// end of outer loop
getch();
return 0;
}
```

Output of the program

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Consider another example:


```
// nested for loop
#include <iostream.h>
#include <conio.h>
int main()
{
for (int i=1; i<=5; i++) //outer loop
{
for (int j=1; j<=i; j++) //inner loop
cout << "$";
cout << endl;
}
getch();
return 0;
}
```

Output of the program

```
$
$$
$$$
$$$$
$$$$$
```

Summary

- A decision statement causes the control to pass to different parts of the program based on the value of an expression. Decision can be made in C++ in several ways.
- The most important decision statements in C++ are if, if-else and switch Statements.
- if statement uses a condition and based on the values of the condition, the body of if is executed. If the condition is true then body is executed otherwise nothing is executed.
- if -else statement selects either the body of if or else based on the result of the condition. If condition is true then body of if is executed and if false then body of else is executed.
- if-else statement can be nested within each other.
- switch statement is used to select from a number of choices and is a better alternative of nested if-else statement.
- Loops are the iterative statements used to repeat a statement or group of statements until a terminating condition is true.
- for loop is the most commonly used iterative statement in C++ which is used in the situation when one knows the number of iterations in advance.

- while loop is similar to if-else structure in which the condition is checked first and then the body is executed.
- In do while structure the body of the loop is executed at least once even if the condition is false.
- Sometimes, the programmers need to shift the control and make jump from one statement of the program to another statement or part. Commonly used jump statements of C++ language are: break and continue.

Exercise

Q.1 Fill in the Blanks.

i. An if statement can include multiple statements provided that they are _____.
ii. The switch statement controls program flow by executing a specific set of statements, depending on _____.
iii. When the value returned by a switch statement expression does not match a case label, the _____ statements within the label execute.
iv. You can exit a switch statement using a(n) _____ statement.
v. Each repetition of a looping statement is called a _____.

Q.2 Select the correct choice for the following Multiple Choice Questions.

i. Find out the error in following block of code.

```
if (x = 100)
cout << "x is 100";
```

a. 100 should be enclosed in quotations
b. There is no semicolon at the end of first line
c. Equals to operator mistake
d. Variable x should not be inside quotation

ii. Looping in a program means
a. Jumping to the specified branch of program
b. Repeat the specified lines of code
c. Both of above
d. None of above

iii. Which of the following is not a looping statement in C++?
a. while
b. until
c. do
d. for

iv. Which of the following is not a jump statement in C++?
a. break
b. goto
c. exit
d. switch

v. Which of the following is selection statement in C++?
a. break
b. goto
c. exit
d. switch

vi. The continue statement
a. resumes the program if it is hanged
b. resumes the program if break was applied
c. skips the rest of the loop in current iteration
d. all of above

vii. Consider the following two pieces of codes and choose the best answer

1.

```
switch (x) {
case 1:
cout <<"x is 1";
break;
case 2:
cout <<"x is 2";
break;
default:
cout <<"value of x unknown";
}
```

2.

```
if (x==1){
cout <<"x is 1";
}
else if (x==2){
cout << "x is 2";
}
else{
cout <<"value of x unknown";
}
```

a. Both of the above code fragments have the same behaviour
b. Both of the above code fragments produce different effects
c. The first code produces more results than second
d. The second code produces more results than first

viii. Observe the following block of code and determine what happens when x=2?

```
switch (x)
{
case 1:
case 2:
case 3:
cout<< "x is between 1 .. 3";
break;
default:
cout<<"x is not within the range, so need to say Thank You!";
}
```

a. Program jumps to the end of switch statement
b. The code after default will run since there is no task for x=2
c. Will display x is between 1 ... 3
d. None of above

ix. Which of the following is false for switch statement in C++?
a. It uses labels instead of blocks
b. we need to put break statement at the end of the group of statement
c. we can put range for case such as case 1..3
d. None of above

Q.3 Write TRUE/FALSE against the following statements.

i. Decision-making structures cannot be nested.
ii. Body of do while loop is executed at least once if the condition is false.
iii. When ***for*** statement begins executing, at the same time the initialization expression is also executed?
iv. ***switch*** statement is a better alternative of nested if-else structure.
v. Syntax of if-else statement is:

```
else {
}
if (condition) {
}
```

Q.4 Write a C++ program that takes two integers and a chararacter representing one of the following mathematical operations: +, -, /, or *. Use a switch statement to perform the appropriate mathematical operation on the integers, and display the result. If an invalid operator is entered then "Error" message should be displayed and the program should exit (use the exit() function).

Q.5 Write a program using *for* loop that prints even numbers from 0 to 20.

Q.6 Write a program using *while loop* that takes an integer for a variable nValue, and returns the sum of all the numbers from 1 to nValue.
Hint: For example, nValue=5 should return 15, which is 1 + 2 + 3 + 4 + 5.

Q.7 What is wrong with the following for loop?

```
// Print all numbers from 9 to 0
for (unsigned int nCount = 9; nCount >= 0; nCount--)
cout << nCount << " ";
```