Name :: Awais Ghafar

ID :: 15269

Dep. :: BScs.

Paper :: Programing fundamental

Teacher :: Sir Dr. Fazal Malik

(Q:NO: 1 Part A)

(IF statement)

An if statement is a Programing conditional statement that, if proved true, Performs a function or displays information. Below is a general example of an if statement, not specific to any Particular Programing language.

```
if (n < 10) {
    Print "Hello John";
}
```

The if statement is used to check a condition and if the condition is true we run a block of statements called the else block)

The else clause is optional.

②

(Q.1, Part A)

(Forms)

Two types of if condition are iF - THEN and IF - THEN ELSE.

```
// IF Condition
# include <iostream>
main ()
{
    if ( Condition)
    {
        // Body of if statement
    }
}

// IF - THEN - ELSE Condition
main()
{
    if ( (condition)
    {
        // Block of code if condition is true
    }
    else {
    // Block of code if condition is false
    }
}
```

③

(Q.NO: 1) B.

```
# include <iostream>
# include <conio.h>
Using namespace std;
int main(){
    int n1, n2;
    cout<<"Enter two number from
            keyboard:\n";
    Cout<<"enter first number\n";
    Cin>>n1;
    Cout<<"enter 2nd number\n";
    Cin >>n2;
    if (n1>=n2)
    {
        cout<<"Largest number is:";
        Cout<< n1;
    }
    else {
    cout<<" Largest number is:";
    cout <<n2;
    }
        return 0;
}
```

(Q. 2. Part A)

(Logical operator;)

A Logical operator is a symbol or word used to connect two or more expression such that the value of the compound expression and on the meaning of the operator.

Common operator. AND. OR and NOT.

<u>AND operator;</u>

The AND operator is a Boolean operator used to perform a logical conjunction on two expressions. Expressions 1 AND expression 2 AND operator returns a value of TRUE if both its operands are TRUE and FALSE.

ii) (OR Operator)

The OR operator is a Boolean operator which would return the value TRUE or Boolean value of 1 if either or both of the operands are TRUE or have Boolean value of 1. The OR operator is considered along with AND and NOT in Boolean algebra.

Not operator:

The boolean not operator is represented with an exclamation sign!

The syntan is pretty simple

1 - result = ! value

The operator accepts a single argument and does the following

true / False

Return the inverse value.

1 - alert (! true) : // false

2 - alert (! 0) : // true

|| (OR operator)

The OR operator represented with ~~respond~~ to vertical line symbols.

1. result = a || b;

There are four possible.

```
alert (true || true);
alert ( False || true);
alert (true || False);
alert ( False || False);
```

&& (AND operator)

AND is presented by &&

1. result = a && b,

```
alert (true && true); //
alert (false && true); //
alert (true && False); //
alert (False && false); //
```

7

(Q: NO: 2 Part B)

```
#include< iostream>
# include <conio.h>
Using namespace std;
main ()
{
    int temp;
    cout <<" temperature is \n";
    Cin >> tmp;
    if (tmp >=40)
    {
     cout <<" its very hot \n";
    }
     else if(tmp>35 && tmp<40)
     {
     cout <<" its tolerable \n";
     }
      else if(tmp >=30 && tmp<=35)
      {
       cout <<" its warm \n";
      }
        else
        {
         cout<<" cool";
        }
}
```

⑧

(Q.NO: 3 A)

Looping.

A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. for example when we are displaying number from 1 to 100 you may want set the value of variable to 1 and display it 100 times increasing its value by (1) on each loop iteration

(While loop in c++)

Loops are used for executing a block of program statements repeatedly until the given loop condition.

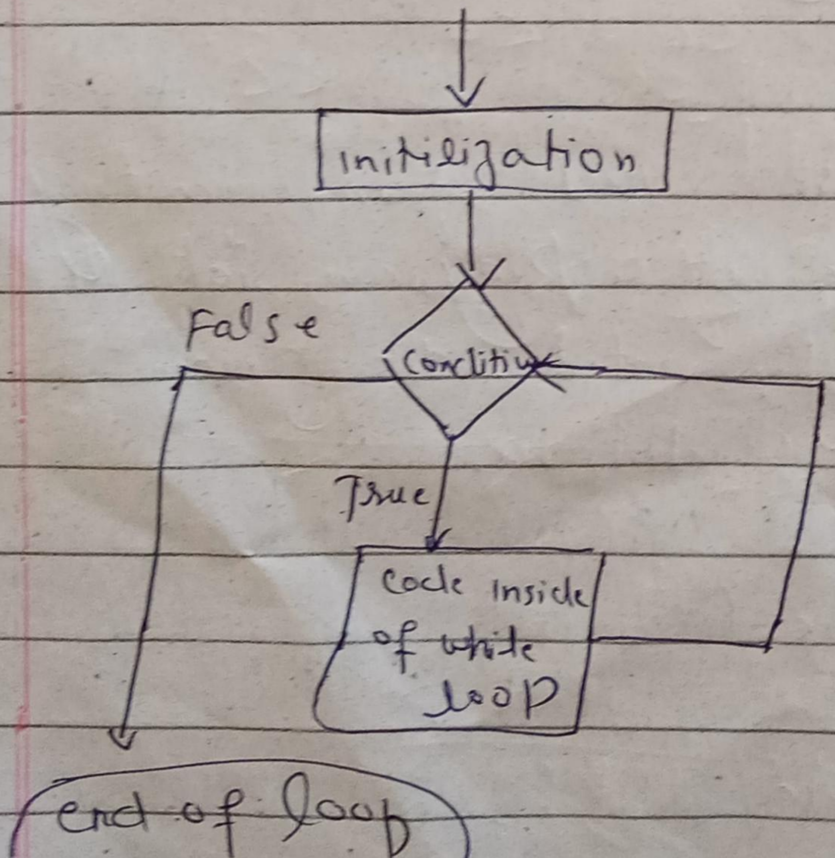
Syntax of while loop:

```
While (condition)
{
   statement(s);
}
```

(Q:3 Part A)

In while loop condition is evaluated first and if it returns true then the the statements inside while loop execute This happens repeatedly until the condition returns false. when condition returns false. the control comes out of loop and jumps to the next statement in the program after while loop.

(Diagram)



10

Q.3 Part A.

(do-while loop)

First the statements inside loop execute and then the condition gets evaluated, if the condition returns true then the control jumps to the 'do' for further repeated execution of it this happens repeatedly until the condition returns false once condition returns false control jumps to the next statements in the program after do while.

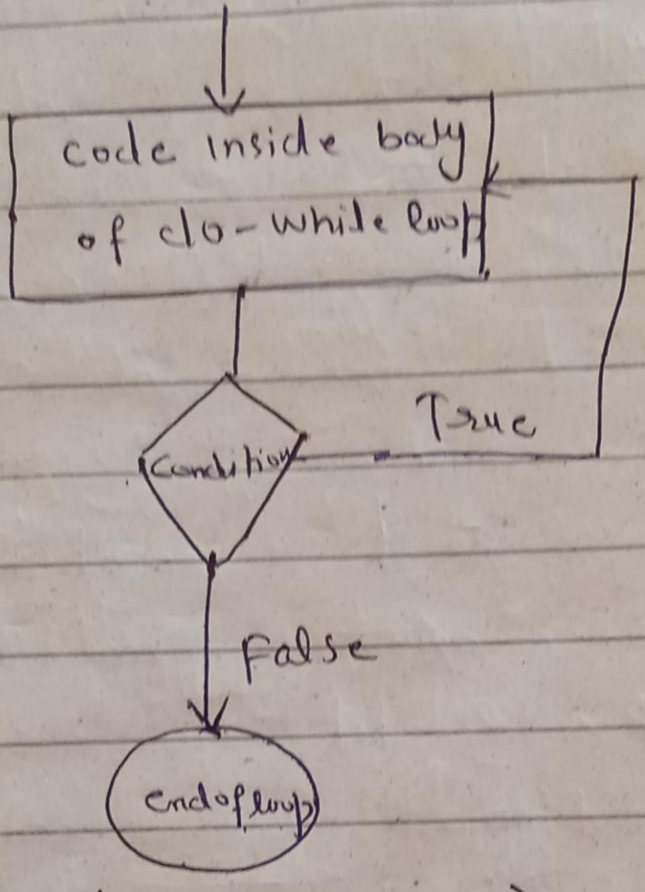
(Syntax)

```
do
{
    statement(s);

} while (condition);
```

Q: 3 part A.

Diagram of Do-while loop.



(For loop)

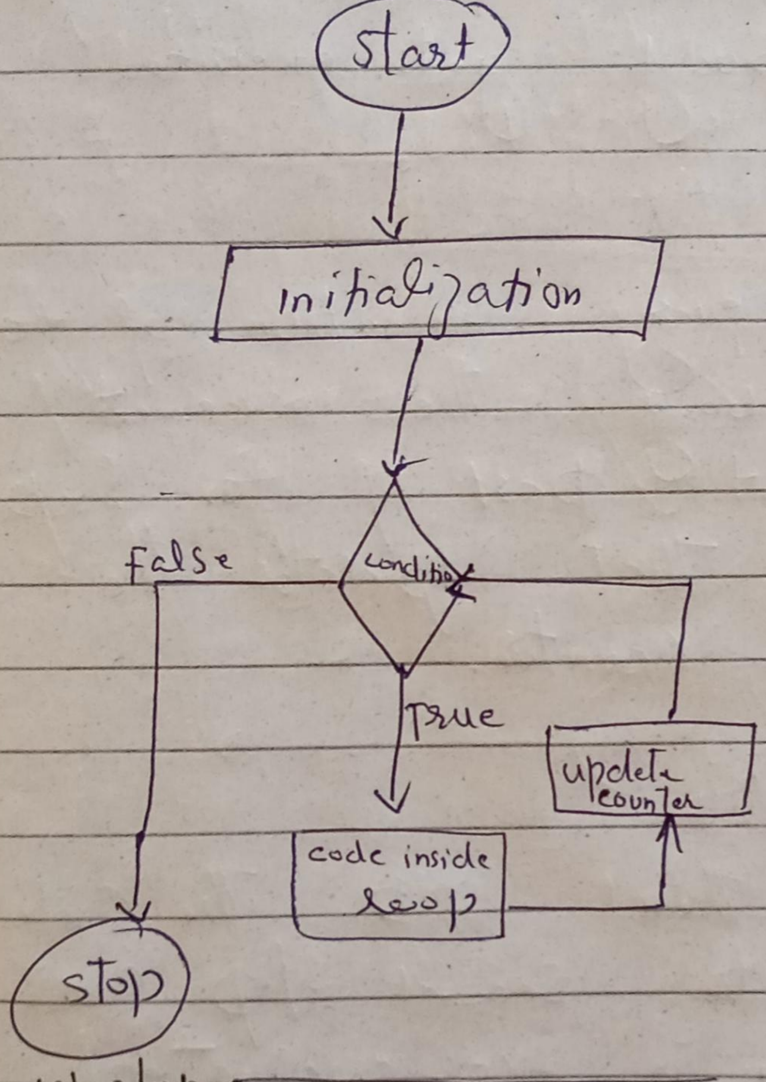
(Synlan)

```
for( initialization; condition;
  increament
{
   C++ statement(S);
}
```

(12)

Q: 3 A part.

(Diagram)



1st step

In for loop initilization happens First and only once which means that the initialization part of for loop only execute once

2nd step

Condition of for loop is evaluated on each loop iteration

Q.3 Part A.

if the condition is true then the statements inside for for loop body gets executed

3rd step:

After every execution of for loop's body the increament / decreament part of for loop executes That updates the loop Counter.

4th step.

After third step, The control jumps to second step and condition is re-evaluated. The steps from second to fourth repeats until the loop condition returns false

(Q: NO: 3 Part B)

```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
    int number;
    cout<<"Enter the number\n";
    cin>>number;
    if (number % 2 == 0)
    cout << number<<" is an
    even number";
    else
    cout << number<<" is an
    odd number";
    return 0;
}
```

15

Q: 4. A.

Break and Continue statement?

Break statement used to "Jump out of a ~~loop~~ switch" statement.

It can also be used to Jump out of loop.

(example)

```
for (int I = 0; I < 10; I++) {
 if (I == 4) {
    break;
 }
  Cout << I << "\n";
}
```

(Continue statement)

The continue statements breaks one iteration (in the loop) if a specified condition occurs and continues with the next iteration in the loop.

(example)

```
for (int i = 0; I <10; I++) {
    if (I == 4) {
        Continue;
    }
    Cout << I << "\n";
}
```

(Q:4 Part B)

```
#include < iostream>
# include < conio.h>
using namespace std;
int main()
{
    int i;
    int sum=0;
    cout<< " The first 10 natural number
     are :\n";
    for (I=1; I<=10; I++)
{
    cout<<i<<" ";
    Sum = Sum+I;
}
   cout << "\nThe sum of
first 10 natural number is \n";
Cout<<sum;
  return;
}
```

Q.NO: 5

(a) C++ Character Set.

In C++ Character is a set of all valid character that can be used in a C++ Program. Character set is used to specify the character or symbols recognized by the language. character set is a set of all valid character that can be used to forms words number and expressions character used for the source program text while the execution character set is the set of character used during the execution of the program. It is not necessary that source character set match and execution character set are same.

This include following character

Q.No.5.

26 lowercase Roman character

a b c d e f g h i j k l m n o p q

26 uppercase Roman character

A B C D E F G H I J K L M N O P Q.

Digits 0-9

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

The graphic ;

! # % ^ &amp ( ) - +

Blank space, Horizental tab. (→)

Example

```
# include< iostream>
using name space std;
int main()
{
char letter, digit, special, white;
cout<<"W3 Add d - C++ Characterset
cout <<" Enter a letter : ";
Cin >> letter ;
Cout << " you entered a letter: ";
Cin >> letter;
Cout <<" you entered a letter' "
<<letter <<"'"<<\n";
```

Q:5

```
cout<<" Enter a Digit: ";
Cin >> digit;
Cout<<"you entred a digit "
<<digit<<" ' " <<"\n";

Cout <<" enter a special char: ";
Cin >> speial;
Cout << "you entered a special charat.."
<<special << " ' " <<"\n";
cout <<"A horizental (\t) tab";
  return;

}
```

Part. B (Constants).

There are two simple ways in C++ to difine Constants.

Using # define preprocessor

Using const keyword.

Example (Defin identifier value)

```
# include <iostream>
Using namespace std;
```

(21)

Q:5

```
# define Length 10
# define Width 5
# define NewLiNE '\n'
int main() {
  int area {
   Int area;
  area = Length # width;
  Cout << area;
  cout << NENLINE
  return 0;
 }.
```

Const. keyword.

```
# include <iostream>
using namespace std;
int main () {
 Const int Length =10;
 Const int WiDTh =5;
 Const Char NENLINE = '\n';
  int area;
  area = Length # Width;
  Cout << area;
  Cout << NENLINE;
  return 0;
}.
```

(Q:5)

(c) variable.

A variable provide us with named storage that our Program can manipulate Each variable in C++ has a specific type which determines the size the size and layout of the variable's memory, the range of values that can be stored with in that memory and the set of operation that can be applied to the variable. The name of variable can composed of letters digits and the underscore character. it must begin with either a letter or an underscore. upper and lower case letter are distict because C++ is case-sensitive.

(Example)

```
# include <iostream>
using namespace std;
// variable declaration;
```

23

```
extern int a, b;
extern int c;
extern float f;
Int main ( ) {
  // Varable definition
 Int a, b;
  int c;
 float f;
 // actual initialization
   a = 10;
   b = 20;
   c = a + b;
   cout << c << endl;
   f = 70.0/3.0;
   cout << f << endl;
   return0;
 }
```

Q: 5.

(Keywords)

A keywords is a reserved word, you can not use it as a variable name, constant name etc.

This list 32 keywords in C++ language which are also available in C language

List.

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

List which not available in 30 C. lang.

asm, dynamic cast, namespace, bool, reinterpret-cast, explicit, new, static cast false, & catch operator, template friend, private class, this, inline, public, throw, const cast, delete, mutable, promoted, true, try, typeid, typename, using, virtual, ~~wo~~ wchar-t.

(Q:5)

(Relational operator).

A relational operator is used to check the relationship b/w two operands. e.g.

// Checks if a is greater then b a > b;

If the relation is true it returns 1 whereas if The relation is false it returns 0.

(operators)

==	is equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to