Experiment No.1
Basic programming constructs like branching and looping
Date of Performance:
Date of Submission:

Aim :- To apply programming constructs of decision making and looping.

Objective :- To apply basic programming constructs like Branching and Looping for solving arithmetic problems like calculating factorial of a no entered by user at command prompt .

Theory :-

Programming constructs are basic building blocks that can be used to control computer programs. Most programs are built out of a fairly standard set of programming constructs. For example, to write a useful program, we need to be able to store values in variables, test these values against a condition, or loop through a set of instructions a certain number of times. Some of the basic program constructs include decision making and looping.

Decision Making in programming is similar to decision making in real life. In programming also we face some situations where we want a certain block of code to be executed when some condition is fulfilled. A programming language uses control statements to control the flow of execution of program based on certain conditions. These are used to cause the flow of execution to advance and branch based on changes to the state of a program.

- if
- if-else
- nested-if
- if-else-if
- switch-case
- break, continue

These statements allow you to control the flow of your program's execution based upon conditions known only during run time.

A loop is a programming structure that repeats a sequence of instructions until a specific condition is met. Programmers use loops to cycle through values, add sums of numbers, repeat functions, and many other things. ... Two of the most common types of loops are the while loop and the for loop. The different ways of looping in programming languages are

- while
- do-while

- for loop
- Some languages have modified for loops for more convenience eg :- Modified for loop in java.

For and while loop is entry-controlled loops. Do-while is an exit-controlled loop.

Code: -

Code for If loop:

```
class SecondProgram
{
 public static void main(String args[])
  {
    int a=5;
    if(a%2==0)
    System.out.println("\n Number is Even\n");
    else
    System.out.println("\n Number is Even\n");
  }
}
```

Output:

```
Microsoft Windows [Version 10.0.22621.2506]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mitanksh\Documents\java practical[1]\AIDS S1 14 Mitanksh>java SecondProgram.java

 Number is Even
```

Code for if Else:

```
public class IfElse
{
    public static void main(String[] args)
    {
        int number=13;
        if(number%2==0){


            System.out.println(number + " is an even number");
        }
        else{
            System.out.println(number + " is an odd number");
        }
```

```
    }
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac IfElse.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java IfElse.java
13 is an odd number
```

Code of Nested if :

```
class NestedIfElse
{
    public static void main (String args[])
    {
    int a=15;
    int b=10;
    int c=20;
    if(a>b)
        {
            if(a>c)
                {
                System.out.println("\na is greater than b and c!");
                }

            else
                {
                System.out.println("\nc is greater than a and b!");
                }
        }
    else
        {
            if(b>c)
                {
                System.out.println("\nb is greater than a and c!");
                }

            else
                {
                System.out.println("\nc is greater than a and b!");
                }

        }
    }
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac NestedIfElse.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java NestedIfElse.java

c is greater than a and b!
```

Code for If – else –if:

```
class ElseIfLadder
{
    public static void main(String args[])
    {
        int marks=90;
        if(marks>=90)
        {
            System.out.println("Student got Grade O as they scored " + marks
+ " marks.");
        }
        else if(marks>=80)
        {
            System.out.println("Student got Grade A as they scored " + marks
+ " marks.");
        }
        else if(marks>=70)
        {
            System.out.println("Student got Grade B as they scored " + marks
+ " marks.");
        }
        else if(marks>=60)
        {
            System.out.println("Student got Grade C as they scored " + marks
+ " marks.");
        }
        else if(marks>=50)
        {
            System.out.println("Student got Grade D as they scored " + marks
+ " marks.");
        }
        else if(marks>=40)
        {
            System.out.println("Student got Grade E as they scored " + marks
+ " marks.");
        }
        else if(marks<40)
```

```
        {
            System.out.println("Student got Grade F as they scored " + marks
+ " marks.");
        }


        else
        {
            System.out.println("Incorrect Input!");
        }
    }
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac ElseIfLadder.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java ElseIfLadder.java
Student got Grade O as they scored 90 marks.
```

Code for Switch Case:

```
class SwitchCase
{
    public static void main (String args[])
    {
    int WeekDay=5;
    switch(WeekDay)
        {
        case 1:
        System.out.println("\nIt's Monday!");
        break;

        case 2:
        System.out.println("\nIt's Tuesday!");
        break;

        case 3:
        System.out.println("\nIt's Wednesday!");
        break;

        case 4:
        System.out.println("\nIt's Thursday!");
        break;

        case 5:
```

```
            System.out.println("\nIt's Friday!");
            break;

            case 6:
            System.out.println("\nIt's Saturday!");
            break;

            case 7:
            System.out.println("\nIt's Sunday!");
            break;

            default:
            System.out.println("\nInvalid Input!");
            break;
            }
        }
    }
```

OutPut:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac SwitchCase.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java SwitchCase.java

It's Friday!
```

Code for While Loop:

```
class Whileloop
{
   public static void main(String args[])
   {
      int M10=1;
      while(M10<=100)
      {
         if(M10%10==0)
         {
            System.out.println(M10);
         }
         M10++;
      }
   }
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac WhileLoop.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java WhileLoop.java
10
20
30
40
50
60
70
80
90
100
```

Code for Do-While Loop:

```
class DoWhileLoop
{
    public static void main(String args[])
    {
        int x=1;
        do
        {
            if(x%25==0)
            {
                System.out.println(x);
            }
            x++;
        }
        while(x<=100);
    }
}
```

OutPut:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac DoWhileLoop.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java DoWhileLoop.java
25
50
75
100
```

Code for For Loop:

```
class Forloop
{
   public static void main(String args[])
   {
      int M20;
      for(M20=1; M20<=100; M20++)
      {
         if(M20%20==0)
         {
            System.out.println(M20);
         }
      }
   }
}
```

Output:

```
C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>javac ForLoop.java

C:\Users\User.DESKTOP-VKOH6B7\Documents\Java Projects>java ForLoop.java
20
40
60
80
100
```

Conclusion:

Comment on how branching and looping useful in solving problems.

- Branching and looping are useful in solving problems because they allow us to control the flow of execution of a program based on certain conditions or repetitions. Branching statements, such as if, else, switch, etc., enable us to choose different paths of execution depending on the outcome of some tests or cases. Looping statements, such as while, for, do-while, etc., enable us to repeat a section of code a number of times or until a condition is met. By using branching and looping, we can make our programs more dynamic, flexible, and efficien