LAB # 9
Looping Statements

Objective:

1. To introduce counter and event controlled loops
2. To work with the while loop
3. To introduce the do-while loop

Theory:

Increment and Decrement Operator

To execute many algorithms we need to be able to add or subtract 1 from a giv-en integer quantity. For example:

```
count = count + 1; // what would happen if we used ==
count += 1;
```

Both of these statements **increment** the value of count by 1. If we replace "+" with "- " in the above code, then both statements **decrement** the value of count by 1. C++ also provides an **increment operator** ++ and a **decrement operator** -- to perform these tasks. There are two modes that can be used:

```
count++; // increment operator in the postfix mode
count--; // decrement operator in the postfix mode
++count; // increment operator in the prefix mode
--count; // decrement operator in the prefix mode
```

The while Loop

A loop is a control struc-ture that causes repetition of code within a program. C++ has three types of loops. The first we will consider is the **while loop**. The syntax is the following:

Sample Program:

```
#include <iostream> using namespace std;
int main()
{
int num = 5; int numFac = 1;
while (num > 0)
{
numFac = numFac * num;
num--; // note the use of the decrement operator
}
cout << " 5! = " << numFac << endl;
return 0;
}
```

Counters

Often a programmer needs to control the number of times a particular loop is repeated. One common way to accomplish this is by using a **counter**. For exam-ple, suppose we want to find the average of five test scores. We must first input and add the five scores. This can be done with a **counter-controlled** loop

Sentinel Values

We can also control the execution of a loop by using a **sentinel value** which is a special value that marks the end of a list of values.

Data Validation

One nice application of the while loop is data validation. The user can input data (from the keyboard or a file) and then a while loop tests to see if the value(s) is valid. The loop is skipped for all valid input but for invalid input the loop is executed and prompts the user to enter new (valid) input.

The **do-while** Loop

A do-while loop is similar to a while loop except that the statements inside the loop body are executed *before* the expression is tested. The format for a single statement in the loop body is the following:

```
do
statement;
while (expression);
```

Lab Task:

Do task 9.1, 9.2 and 9.3 given in file attached on LMS and attach your source code and output window with this file. And write your observation with lab task.

Post Lab:

Write a C++ program that ask a user any integer number and print the sum of all natural numbers up till that number using do while loop.

Sample Run:

```
Enter n:7
Sum = 28
```

Learning Outcomes:

Upon successful completion of the lab, students will be able to:

LO1: To work with looping statements using while and do while loops.