LAB NO: 4

CONTROL STRUCTURES - LOOPING

Objectives:

In this lab, student will be able to:

1. Write and execute C++ programs using 'while' statement
2. Write and execute C++ programs using 'do-while' statement
3. Write and execute C++ programs using 'for' statement

Introduction:

- Iterative (repetitive) control structures are used to repeat certain statements for a specified numbe of times.
- The statements are executed as long as the condition is true
- These types of control structures are also called as loop control structures
- Three kinds of loop control structures are:
    - while
    - do-while
    - for

C++ looping control structures:

While loop:

```
      while(test condition)
    {
      body of the loop
    }
```

Do-while loop:

```
do
{
body of the loop
}
while (test condition);
```

For loop:

```
for (initialization; test condition; increment/decrement)
{
body of the loop
}
```

Solved exercise

[Understand the working of looping with this illustrative example for finding sum of natural numbers up to 100 using ***while*** and ***do-while*** statements]

Using ***do-while***

```
#include <iostream.h>
void main()
{
int n;
int sum;
sum=0; //initialize sum
n=1;
do
   {
sum = sum + counter;
counter = counter +1;
   } while (counter < 100);
cout<<sum;
}
```

Using ***while***

```
#include <iostream.h>
void main( )
{
int n;
int sum;
sum=0; //initialize sum
n=1;
while (n<100)
   {
      sum = sum + n;
       n = n +1;
   }
cout<<sum;
 }
```

Lab exercises

With the help of iterative (looping) control structures such as ***while***, ***do-while*** and ***for*** statements,

Write C++ programs to do the following:

1. Reverse a given number and check if it is a palindrome or not.
   [Ex: 1234, reverse=$4*10^3 +3 * 10^2 + 2 * 10^1 + 1 * 10^0$ =4321]

2. Generate prime numbers between 2 given limits.

3. Check if the sum of the cubes of all digits of an inputted number equals the number itself (Armstrong Number).

4. Generate the multiplication table for ‘***n***’ numbers up to ‘***k***’ terms (using nested for loops).

[Hint: 1 2 3 4 5 .... k
2 4 6 8 10 ....2*k
....
n................. n*k]

Additional exercises

1. Check whether a given number is perfect or not.
[Hint: Sum of all positive divisors of a given number excluding the given number is equal to th
number] Ex: 28 = 1+ 2 + 4 + 7 + 14 = 28 is a perfect number

2.// Evaluate the sine series, sin(x)= x- x^3/3! + x^5/5!–x^7/7!+ .... to n terms.

3.// Check whether the given number is strong or not.
[Hint: Positive number whose sum of the factorial of its digits is equal to the number itself]
Ex: 145 = 1! + 4! + 5! = 1 + 24 + 120 = 145 is a strong number.

4. Find out the generic root of any number.
[Hint: Generic root is the sum of digits of a number until a single digit is obtained.]
Ex: Generic root of 456 is 4 + 5 + 6 = 15 = 1+5 = 6

5.// Generate Floyd's triangle using natural numbers for a given limit N.
[Hint: Floyd's triangle is a right angled-triangle using the natural numbers]
Ex: Input: N = 4
Output:

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