National University of Computer and Emerging Sciences

Lab Manual # 7

Programming Fundamentals

(Section BCS- IG)

Course Instructor	Mr. Raziuddin
Lab Instructor(s)	M.Naveed Ms. Sonia Anum
Section	BSE-1G
Semester	Fall 2021

Department of Computer Science

FAST-NU, Lahore, Pakistan

Lab Manual

Objectives

The objectives of this lab are to cover the following:

- user-defined functions (page 320 of textbook)
- parameter passing to functions by value
- nested loops (page 291 of textbook)

Function:

A function is a group of statements that is given a name, and which can be called from some point of the program. The most common syntax to define a function is:

```
type name ( parameter1, parameter2, ...)
 {
        statements
}
```

Where:

- **type** is the datatype of the value returned by the function.
- **Name** is the identifier by which the function can be called.
- **Parameters** (as many as needed): Each parameter consists of a datatype followed by an identifier, with each parameter being separated from the next by a comma. Each parameter looks very much like a regular variable declaration (for example: int x), and in fact acts within the function as a regular variable which is local to the function. The purpose of parameters is to allow passing arguments to the function from the location where it is called from.
- **Statements** is the function's body. It is a block of statements surrounded by braces { } that specify what the function actually does.

Example:
```
// function example
#include <iostream>
using namespace std;

int addition (int a, int b)
{
        int r;
        r=a+b;
        return r;
}
int main ()
{
        int z;
        z = addition (5,3);
        cout << "The result is" << z;
}
```

Output:
```
The result is 8
```

Exercise 1:

Write a program that asks the user to enter an item's wholesale cost and its markup percentage. It should then display the item's retail price.

For example:
If an item's wholesale cost is 5.00 and its markup percentage is 100%, then the item's retail price is 10.00.
If an item's wholesale cost is 5.00 and its markup percentage is 50%, then the item's retail price is 7.50.

The program should have a function named calculateRetail that receives the wholesale cost and the markup percentage as parameters and returns the retail price of the item.
Input Validation: Do not accept negative values for either the wholesale cost of the item or the markup percentage.

Exercise 2:

Write C++ program that will ask the user to enter the width and length of a rectangle and then display the rectangle's area. The program calls the following functions, which have not been written:

- **getLength:** This function should ask the user to enter the rectangle's length and then return that value as a double .
- **getWidth:** This function should ask the user to enter the rectangle's width and then return that value as a double .
- **getArea:** This function should accept the rectangle's length and width as arguments and return the rectangle's area. The area is calculated by multiplying the length by the width.
- **displayData:** This function should accept the rectangle's length, width, and area as arguments and display them in an appropriate message on the screen.

.

Exercise 3 (function called within loop):

When an object is falling because of gravity, the following formula can be used to determine the distance the object falls in a specific time period:
d =1/2*g*t2
The variables in the formula are as follows: d is the distance in meters, g is 9.8, and t is the amount of time, in seconds, that the object has been falling.
Write a function named **DistanceFallen** that accepts an object's falling time (in seconds) as an argument. The function should return the distance, in meters, that the object has fallen during that time interval. Write a program that demonstrates the function by calling it in a loop that passes the values 1 through 10 as arguments and displays the return value.

Exercise 4 (nested loop within function):

Write a function called **displayShape** that accepts size and shapetype as parameters and displays a square or a triangle of the required size.

So e.g. if the function is called with the following parameters:

displayShape('S', 4); a square of size 4 is displayed as follows

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

And if the function is called with the following parameters:
displayShape('T', 4); a triangle of size 4 is displayed as follows

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