LAB # 11

Functions

Objective:

1. To work with void functions (procedures) that have no parameters.
2. To introduce the concept of functions that return a value
3. To work with void functions (procedures) that have pass by value and pass by reference parameters.
4. To understand the difference between static, local and global variables

Theory:

One-Dimensional Arrays

A key element of structured (well organized and documented) programs is their modularity: the breaking of code into small units. These units, or modules are called functions. There are two types of functions; void functions and functions that return a value.
In simple programs most functions are called, or invoked, by the main function.
Calling a function basically means starting the execution of the instructions contained in that module. Sometimes a function may need information “passed” in order to perform designated tasks.

```
#include <iostream>
using namespace std;
void printDescription(); // Function prototype
int main()
{
cout << "Welcome to the Payroll Program." << endl;
//printDescription(); // Call to the function
cout << "We hoped you enjoyed this program." << endl;
return 0;
}
////////////////////////////////////////////////////
///////////////////////////////////////////////////
void printDescription() // The function heading
{
cout << "***************************************************"
<< endl << endl;
cout << "This program takes two numbers (pay rate and hours)"
<< endl;
cout << "and outputs gross pay. " << endl;
cout << "***************************************************"
<< endl << endl;
}
```

Pass by Value
The actual parameters pass their values to their corresponding formal parameters. This is called pass by value.
Pass by Reference

Lab Task:

Do task 11.1, 11.2, 11.3 and 11.4 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:

1. Write a program that will read two floating point numbers (the first read into a variable called first and the second read into a variable called second) and then calls the function swap with the actual parameters first and second. The swap function having formal parameters number1 and number2 should swap the value of the two variables.

Sample Run:

```
Enter the first number Then hit enter
80
Enter the second number Then hit enter
70
You input the numbers as 80 and 70.
After swapping, the first number has the value of 70 which was the value of the
second number
The second number has the value of 80 which was the value of the first number
```

2. Write a program that will convert miles to kilometers and kilometers to miles. The user will indicate both a number (representing a distance) and a choice of whether that number is in miles to be converted to kilo-meters or kilometers to be converted to miles. Each conversion is done with a value returning function. You may use the following conversions.

Sample Run:

```
Please input
1 Convert miles to kilometers
2 Convert kilometers to miles
3 Quit
1
Please input the miles to be converted
120
120 miles = 193.2 kilometers
Please input
1 Convert miles to kilometers
2 Convert kilometers to miles
3 Quit
2
Please input the kilometers to be converted
235
235 kilometers = 145.935 miles
Please input
1 Convert miles to kilometers
2 Convert kilometers to miles
3 Quit
3
```

Learning Outcomes:

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

LO1: To work with looping and nested looping statements using void functions as well as functions that return value.