60-140 INTRODUCTION TO ALGORITHMS AND PROGRAMMING I
Dr. Ziad Kobti

Lab 6: More on functions

Objectives:
1. Learn the structure of a function: prototype, definition, documentation, and invocation.
2. Understand the top-down process and why we use functions. (Divide and conquer strategy)
3. Understand software re-use. (Make your functions portable so you can reuse them elsewhere)
4. Nested functions (Hierarchy of functions invocations) and call-stack.

Instructions:
Recall in the previous lab you learned how to create a directory, change into it, and view its contents. (i.e. the UNIX commands you need to understand here are `mkdir` *`directory_name`*, `cd` *`directory_name`*, and `ls`)

In this lab you are asked to create one complete C program that includes several functions, but all inside the same source code file named **Lab6.c** in a directory called **lab6** which should be located inside the **cs140** directory you have already created in the first lab. The reason for this is to keep your files organized.

At the end of this lab class, you will have 1 source code files saved in a folder called lab6 under the cs140 folder. Recall that for the purpose of evaluating and grading your work by the GA's, you would need to create one script file which holds the source code, compilation result and the output of all programs. The process of creating such a file is described below:

```
>> script lab6.txt
>>  cat  lab6.c    // will show and add the source code for the
question to the script file
>>cc lab6.c       // will compile lab6.c
>>./a.out         // will run the program and appends its output to
lab6.txt file
>>exit            //closes the script file. Your work is ready to be
graded!
```

Submission:
Your lab is graded either at the end of your current lab class (or at the very beginning of your next regular lab class without penalty). Late labs without a valid excuse (e.g. illness) receive 0. You are to present your code on the computer in the lab to your lab instructor to receive a grade as follows:

0= not satisfactory or no documentation; 1=incomplete; 2=complete and well documented

Lab question

You are to write a single c program that displays a simple menu with a few menu items. Each menu item is associated with a C function that performs a certain process as elaborated below.

Hint: the structure for this C program is as follows

```
int main()
{
      MenuController();

      Return 0;
}
```

List of functions: (recall a complete function includes: documentation, prototype and definition)

GetNumber()
This function prompts the user to input an integer number between integer min and max and returns a valid entry between min and max inclusive; both min and max are integer function parameters. It is obvious that this function validates the input before returning the control to its invoker by repeatedly prompting for a valid number within the range.

DisplayMenu()
This function displays the menu of the possible actions as follows

```
ACTION LIST MENU
----------------

   1. Compute Factorial of n
   2. Compute Fibonacci of n
   3. Compute Sum_Odd of n
   4. Exit
```

MenuController
This function invokes functions GetNumber and DisplayMenu. Based on the user selection, it invokes the corresponding function to perform the task. This process will be performed repeatedly until the user chooses to exit.
The menu selection decision structure should be implemented using a *`switch`* statement based on the choice made by the user.

ComputeFactorial
This is a <u>non-recursive</u> function that accepts an integer n as input parameter and computes and returns n!
Hint: n!= n*(n-1)*(n-2)*…1
Example: 5!=5*4*3*2*1=120

ComputeFibonacci
This is a non-recursive function that accepts an integer n as input parameter and computes and returns the Fibonacci value of it.
Hint: fib(n)=fib(n-1)+fib(n-2)
    fib(1)=1
    fib(0)=0
Example:
fib(5)=fib(4)+fib(3)=3+2=5

SumOdd
This function accepts an integer n as input and computes and returns the total of all odd numbers within the range of 1..n (including n itself if n is odd)
Example:
SumOdd(5)=1+3+5=9

Sample output of the program:

```
ACTION LIST MENU
----------------

1. Compute Factorial of n (n!)
2. Compute Fibonacci of n
3. Compute SumOdd of n
4. Exit

Please select one action from the above list? 5
Your entry was invalid!
Please select one action from the above list? 1
The factorial of 6 (6!) = 720
Please select one action from the above list? 2
The Fibonacci of 6 = 8
Please select one action from the above list? 3
The total of odd numbers between 1 and 6 = 9
Please select one action from the above list? 4
Goodbye!
```