IC 150 P Computation for engineers lab

Lab assignment sheet no: 4, Odd semester, 2016

Functions and Pointers

Prepared by: Timothy A. Gonsalves, Maben Rabi

Objective for this lab session

- To learn to break up a computational task into smaller units, and to write a series of appropriate functions. To Write functions that can be called recursively.
- To experiment with pointers.

Task one

Write a program to find the minimum of a given function, by using the Golden section method from the previous assignment. Write two functions

`void goldenSection(float* , float* , float* )` that when called will shrink the initial interval of uncertainty by the Golden ratio $\left(\sqrt{5}-1\right)/2$. The function will achieve this by carrying out exactly one round of the Golden section method. Please note that you can pass to this function only pointers corresponding to the addresses of: (1) the left end of the interval, (2) the right end of the interval, and (3) the minimum of function values evaluated thus far.

`float evaluateObjectiveFunction(float )` that when called will return the value of the chosen objective function. You may choose your favourite unimodular function, or the function IAE (ζ) from the previous assignment.

Task two

Part A: Write a program to compute the factorial of a given number. Write 3 functions, and write them in such a way that `Fact(n)` is computed by calling itself recursively. The three functions to be written are:

`int Input()`: reads a number from the keyboard and returns it. The function should ensure that the number returned is non-negative.

`int Fact(int n)`: computes the factorial of its argument.

`void Output(int n, int f)`: takes the number and its factorial and prints them.

Part B: Similar to above, bu without calling any function recursively: compute the GCD of two given numbers.
Part C: In the `Fact()` function, declare a local variable f to hold the computed value (as shown in the IC 150 class lecture lecturess). At the end of `Fact()`, print the value of this variable, and its address. Call `Fact(5)`, and draw the memory diagram showing the call stack.

Task three

Define trivial functions `F()`, `G()` and `H()`. Each of these functions takes 2 arguments and each has 2 local variables with the same names `a` and `b`. The `main()` function also has 2 variables `a` and `b` . Specify some behaviour for `F()`, `G()` and `H()` such that the values of the variables a and b in each of these are different. `main()` calls `F()` which calls `G()` which calls `H()`. At the end of each function, print the function name and the values of its local variables `a` and `b`. At the start of each function, print the address of its local variable a, and the addresses of its arguments. Using these, draw a memory diagram showing the call stack.

`G()` should pass the address if its `b` to `H()`. `H()` modifies this reference variable. In `G()`, print the value of b before and after the call to `H()`.