Mahesh Gautam

CS 315

Fall 2014

Lab 01

Exercises:

1. All values resulting from the equation **‘3n+1’** will halt the call to function f(n).

Example:

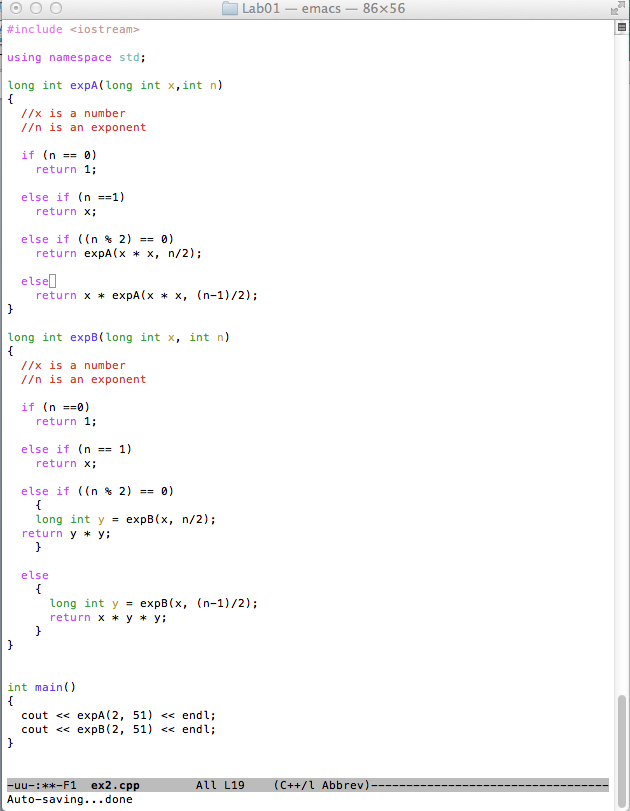
When n = 1; 3n+1 = 4

So, return f(n-3) + 1 would return

f(1) + 1

and this will halt the program.

**Please proceed to next page for Exercise #2**

1. Code:

**251 = 2251799813685248**

3)

**a) f (1000)**

= 1 + f(500)

= 1 + 1 + f(250)

= 1 + 1 + 1 + f(125)

= 1 + 1 + 1 + 2 + f(62)

= 1 + 1+ 1 + 1 + 2 + f(31)

= 1 + 1 + 1 + 1 + 2 + 2 + f(15)

= 1 + 1 + 1 + 1 + 2 + 2 + 2 + f(7)

= 1 + 1 + 1 + 1 + 2 + 2 + 2 + 2+ f(3)

= 1 + 1 + 1 + 1 + 2 + 2 + 2 + 2+ 2 + f(1)

= 1 + 1 + 1 + 1 + 2 + 2 + 2 + 2+ 2 + 0

**= 14**

**b) f (2k)?**

**= K**

The answer is ‘K’ because f is always even and also ‘k’ is a power of 2.

So, 2 keeps multiplying itself times ‘K’.

**c) f (2k-1)?**

= 2K -2

**= 2(k-1)**

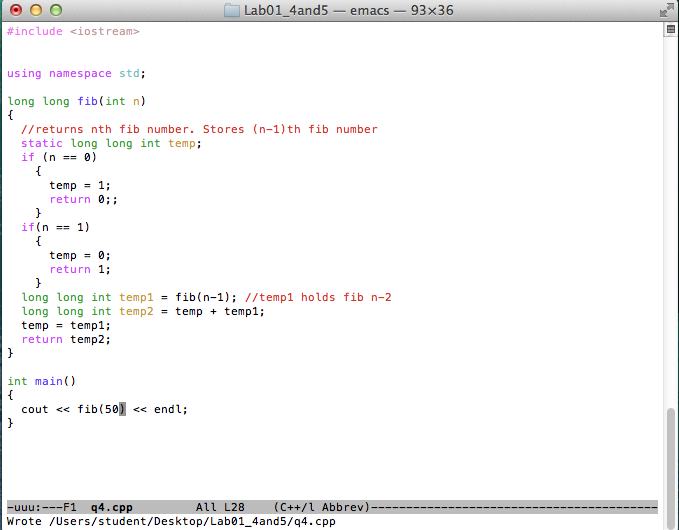
4) F(n) = F(n-1) + F(n-2)

* How many times is a call for F(1) made when computing F(10)?

F(1) is called **55** times when computing F(10)

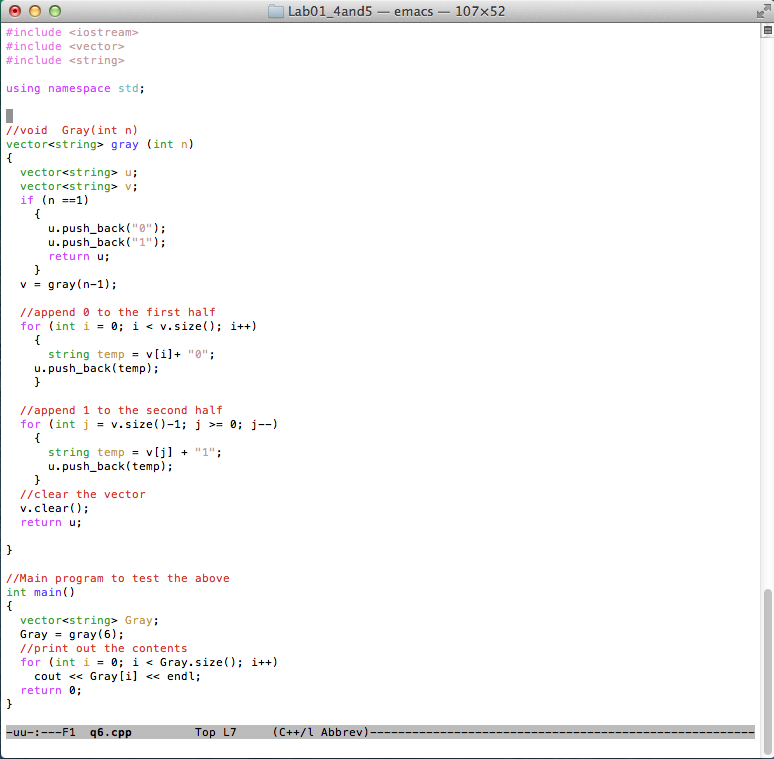
* Write a recursive function for computing the nth Fibonacci number that avoids redundancy.

= see code below (screenshot)



* Use your function to compute f(50).

**f (50) = 12586269025**

5) Code:  


Output when n = 6

