LAB #7

Each lab will begin with a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstances, contact your lab TAs and Jennifer Parham-Mocello.

Reminder: All of our labs involve paired programming. You do not have to keep the same partner for each lab, but you MUST work with someone in each lab!!! **First, find a partner for this lab.** It can be the same partner from the previous lab or a different partner.

Take the first 30 minutes of lab to work on lab #6 from last week!!!

(7 pts) Iteration vs. Recursion

You will practice writing and timing iterative vs. recursive functions. You will use the following code to time your iterative versus recursive solution to the following problem.

```
#include <sys/time.h>
#include <cstdlib>

using sdt::cout;
using std::endl;

int main() {
     typedef struct timeval time;
     time stop, start;

     gettimeofday(&start, NULL);
     //Time your iterative or recursive function here.

     gettimeofday(&stop, NULL);

     if(stop.tv_sec > start.tv_sec)
        cout << "Seconds: " << stop.tv_sec-start.tv_sec << endl;
     else
        cout << "Micro: " << stop.tv_usec-start.tv_usec << endl;

      return 0;
}
```

First, you will write an function called, fib_iter(), that has one parameter, n, of type int, and it returns the nth Fibonacci number, F_n. The Fibonacci numbers are F_0 is 0, F_1 is 1, F_2 is 1, F_3 is 2, F_4 is 3, F_5 is 5, F_6 is 8, etc.

$F_{i+2} = F_i + F_{i+1}$ for $i = 2, 3, \ldots$; where $F_0 = 0$ and $F_1 = 1$

In the iterative function, you should use a loop to compute each Fibonacci number once on the way to the number requested and discard the numbers when they are no longer needed.

Next, write a recursive function called, fib_recurs(), that also takes one parameter, n, of type int, and returns the nth Fibonacci number, F_n.

After each function, print the time it takes for the nth Fibonacci number to be calculated. **Time the iterative and recursive solutions for finding the 1st, 5th, 15th, 25th, and 45th Fibonacci numbers**. Determine how long each function takes, and compare and comment on your results.

Show your TA a trace through the recursive solution for n=5.

Application: You're standing at the base of a staircase and are heading to the top. A small stride will move up one stair, a large stride advances two. You want to count the number of ways to climb the entire staircase based on different combinations of large and small strides. For example, a staircase of three steps can be climbed in three different ways: via three small strides or one small stride followed by one large stride or one large followed by one small.

How could you apply the Fibonacci number series to this problem? What are the base cases for counting the ways you can climb stairs by going one stair or two stairs at a time? How many different ways can you climb 4 stairs? 5 stairs?

(3 pts) Static 1-d arrays: C-style Strings

Change your implementation from lab #6 to use C-style strings, instead of C++ strings. A C-style string is a string of characters ended by the null character, '\0'. Since you don't know how big the message will be, you need to declare a character array large enough to hold a sentence, usually 256 characters will do! This means that the string can hold a maximum of 255 characters, plus one null character. ☺

```
char mssg[256];
```

Create two more C-style strings for your search and replace strings too, but these can be smaller if you would like. Your cin and/or getline calls do not have to change, and these functions will automatically insert a null character at the end of the characters read from the user. Since you are using a C-style string now, you will need to use the cstring (or string.h) library, instead of string.

http://www.cplusplus.com/reference/cstring/?kw=cstring

Notice, there are still functions for copying, searching, and finding the length of C strings, but replacing a string might not be as easy!!! You function prototypes from lab #6 will change to use C strings, rather than C++ strings. Remember, C strings use character arrays, which are pointers. This means you do not need to pass by reference anymore. By default, the contents of an array can change inside a function, when you pass the name of the array as an argument to the function. You will need to change your function prototypes and definitions to character pointers or character arrays:

```
void get_sentence(char *);
bool determine_palindrome(char *);
void get_search_replace_strings(char *, char *);
int search_replace(char *, char *, char *);

OR

void get_sentence(char []);
bool determine_palindrome(char []);
void get_search_replace_strings(char [], char []);
int search_replace(char [], char [], char []);
```

Now, re-write these functions to work with C strings, instead of C++ strings.