Lab 6

Each lab will begin with a recap of last lab and 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. For non-zero labs, 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 circumstance, contact your lab TAs and Jennifer Parham-Mocello.

Pair Programming: In this lab, you can choose a partner for pair programming. **You must be checked off together. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, you will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: Pair Programming Student Handout

(6 pts total) Design First (3 pts), Then Write Code (3 pts)!!!

To help you practice functions for Assignment #4, you will write a short program that asks the user to enter a string (**get_string**()), makes a copy of the string, takes the copy and changes all non-space letters to dashes (**set_replace_string**()), and gets a letter from the user to search and replace returning the number of letters found (**get_search_replace**()).

You should design first…

You can have more functions, but you must have at least the three below with the EXACT function prototypes.

Each of your functions should have the proper function headers/descriptions

```
void get_string(string *);
void set_replace_string(string, string *);
int get_search_replace(string, string &);
```

Write the function headers/descriptions for each of the functions above, as well as all the functions you create! This includes information about parameters, return values, and pre/post conditions.

Design – Give as much detail as possible for the main function and all the functions above.

- Why do the function prototypes have the specific parameter types on them?
- Why do the functions have void or a return value?
- How do all the functions interact together?

Testing – Provide testing values with expected results.

- What do you plan to use as bad values? Good values?
- What do you expect to happen with bad values? Good values?

If any of your functions are more than 10 lines of code, what can you do to make it smaller? If you are having difficulty thinking of how to make it smaller, then ask a TA to help you!!!

STOP!!! Get checked off by a TA before beginning to implement. This will help with logic and function mistakes**.**

Now, INCREMENTALLY write your functions and program!!!!

(4 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.

Fibonacci 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?