Lab 6

Get checked off for up to 3 points of incomplete work from the previous lab within the first 10 minutes of the lab.

In this lab, you can form a group of 2-3 individuals. **You must be checked off together as a group at the end of the lab.** Although you perform tasks as a group, ensure that you understand the work and ask questions to TAs as needed.

Pass by References

Let's take a minute to revisit the second part of Lab 5 where a function called **swap_sentence(string sen1, string sen2)** that took two string inputs as parameters and swapped them.

Example code:

```
void swap_sentence(string sen1, string sen2){
     // swap functionality implemented by you…

     cout << "In swap function …" << endl;
     cout << "sen1: " << sen1 << endl;
     cout << "sen2: " << sen2 << endl;
}
int main() {
      string sentence1, sentence2;

      cout << "Enter sentence 1: ";
      getline (cin, sentence1);
      cout << "Enter sentence 2: ";
      getline (cin, sentence2);

      swap_sentence(sentence1, sentence2);

      cout << "In main function …" << endl;
      cout << "sentence1: " << sentence1 << endl;
      cout << "sentence2: " << sentence2 << endl;

      return 0;
}
```

Implementation (3 pts)

Now, let us implement the swap_sentence(), so it modifies the string parameter directly.

Based on the function call above, notice that swap_sentence() is a void function because we don't do anything with the function call (it is on a line by itself). It has two strings as arguments in the function call. Therefore, the prototype for this function is as follows. **void swap_sentence(string sen1, string sen2);**

What did you notice?

From the implementation, you may have already noticed that we can use the strings inside the function, but the two strings are not swapped after calling the function. In other words, the values of sentence1 and sentence2 are not changed inside the swap_sentence() function. In order to change the value of the strings inside the function, we need to add an ampersand (&) in front of the parameter.

void swap_sentence(string &sen1, string &sen2);

Before moving on, here are a couple of questions to think of:

1. Can we change the value of the string inside the function if we match up the parameter and argument names? For example, change the function prototype to:

   void swap_sentence(string sentence1, string sentence2);

2. What is indeed passed into the function if an ampersand (&) is added in front of the parameter?
3. Can we change the value of the string inside the function if we change the function prototype to:

   void swap_sentence(string *sen1, string *sen2);

What else needs to be changed when we make the function call? Inside the function? 4. What is the difference between an ampersand (&) and an asterisk (*) added in front of the parameter? Use a diagram to explain.

Understand Recursion (1 pt)

Now, let's think of something more interesting. Read through the following code:

```
int factorial (int n); //line 1

int main(){
     int n;
     cout << "Enter a positive integer:";
     cin >> n;

     cout<< "Factorial of " << n << "is " << factorial(n);

     return 0;
 }

int factorial (int n) {
     if (n==1)
          return 1;
```

```
        else
             return n * factorial (n - 1);
    }
```

What if you have a function calling itself? As we did above, use a diagram to trace the code, line by line, and explain what is happening behind the scene.

In addition, answer the following questions:

1. What if line 1 is missing? Explain why.
2. What if “`if (n==1) return 1;` ” is missing? Explain why.
3. There is an int variable n in main function, and an int variable n in factorial function, are they the same variable?

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 std::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;
     }
```

(3 pts) First, you will write a function called, `fib_iter()`, that has one parameter, n, of type int, and it returns the nth Fibonacci number, Fn. The Fibonacci numbers are F0 is 0, F1 is 1, F2 is 1, F3 is 2, F4 is 3, F5 is 5, F6 is 8, etc.

Fi+2 = Fi + Fi+1 for i = 2, 3, …; where F0 = 0 and F1 = 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.

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

Some self - practice (You don't need to be checked for this):

1. 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.
2. **Trace through the recursive solution for n=5 to understand how values change on stack**

Show your completed work and answers to the TAs for credit. You will not get points if you do not get checked off!