Name: (as it would appear on official course roster)
UCSB email address: **@ucsb.edu**
Lab Section Time:
Optional: name you wish to be called if different from above
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone")

Homework 09: Recursive Functions

Assigned: *Tuesday, December 8st, 2020*
Due: *Monday, December 11th, 2020 by 11:59 PM*
Points: *100*

- You may collaborate on this homework with AT MOST one person, an optional "homework buddy".
- MAY ONLY BE TURNED IN ON **GRADESCOPE as a PDF file**. Instructions on How to Submit (applicable to ALL homework assignments in this class) are on Piazza.
- There is NO MAKEUP for missed assignments.
- We are strict about enforcing the LATE POLICY for all assignments (see syllabus).
- **IMPORTANT:** If you use code techniques we have NOT covered in class, you will **get a zero grade** on that problem. If you cheat, or have someone else do your work, you will **get an F in the class**.
  Only use the space provided for answers. Use clear and clean handwriting (or typing).

Reading: Ch. 14

1. (3 pts) How does a recursive function know when to stop recursing?

2. (4 pts) What is stack overflow? When can it occur? What are the consequences?

3. (3 pts) What is a LIFO scheme and how does it relate to stacks?

4. (15 pts) Write a definition of a ***recursive function*** that finds the sum of the odd integers in the first N numbers. Example, if N = 8, then the sum is:
(1 + 3 + 5 + 7) = 16.

```
int sumOdds(int  N)
{




}
```

5. (25 pts) Write a definition of a ***recursive function*** that finds the n^{th} element in the following arithmetic numerical sequence: **3**, **11**, **27**, **59**, **123**, **…**
   I've started the program for you below, but it is missing the definition.
   Hint: First, figure out the recursive pattern as a linear equation, i.e. $\mathbf{a_n = M.a_{n-1} + N}$.
   You also have to identify the base case.
   Example outputs would look like this (there is no repeating loop – these are 2 separate runs):

```
Which element of the sequence would you like to know? 4
Element number 4 in the sequence is 59.

Which element of the sequence would you like to know? 7
Element number 7 in the sequence is 507.
```

```
#include <iostream>
using namespace std;
int RecursiveFunc(int num);
int main()
{
  int elementN;
  cout << "Which element of the sequence would you like to know? ";
  cin >> elementN;
  cout << "Element number 4 in the sequence is " << RecursiveFunc(elementN)
<< endl;
  return 0;
}

//DEFINITION HERE:
```

6. (25 pts) The Fibonacci sequence is defined as a numerical sequence of integers that are the sum of the previous 2 integers. Starting with 0 and 1, the sequence thus becomes: 0, 1, 1, 2, 3, 5, 8, 13, etc…

   Write the definition of 2 functions: one recursive called **`Fibo`** (it finds the ***N***th element in a Fibonacci series) and one non-recursive called **`SFS`** that calls **`Fibo`**.

   `SFS` has an integer argument **N**. The pre-condition is that **N** is assumed to be smaller than 256. The post-condition is that **`SFS`** prints out all the squares of the Fibonacci sequence of the first **N** elements. For example, calling this line in **main** (just like this):

   ```
   SFS(7);
   ```

   will print out to std.out: **`0 1 1 4 9 25 64`**
   (assume you can use `<cmath>`)

7. (25 pts) Write a definition of a recursive function that counts the number of the letter ‘a’ or ‘A’ (i.e. either upper-case or lower-case) in a string. Specifically, given a string variable, **sentence**, when we pass that into a function **CountA(sentence)**, the function returns an integer that’s a count of the number of the letter ‘a’ or ‘A’ in the variable **sentence**. This function must be a recursive one and cannot contain a loop.