# CSUC CSCI-311 - Algorithms and Data Structures
## Lab Assignment 2

Goal: 1) To start thinking about recursion. 2) To practice writing recursive functions in C++.

Submission: C++ solutions to these problems should be written in a file called lab_2.cpp (a skeleton C++ program file as well as the test cases are available on the corresponding Canvas link) and the file should be submitted on inginious (https://inginious.csuchico.edu/).

Coding Style: Note that your submission should use good C++ coding style and should include appropriate comments (for readability and maintainability). Specifically, your code must follow common C++ coding conventions for Naming, Indentation, and Comments. Points will be deducted if these are not present.

Collaboration: There will be time in lab to discuss these problems in small groups and I highly encourage you to collaborate with one another outside of class. However, you must write up your own solutions **independently** of one another. In addition, do not post solutions in any way! Also, please include a list of the people you work with in a comment section at the top of your submission.

Have fun!

| | |
|---|---|
| Assignment Date: | **Sep 09, 2024** |
| Due Date: | **11:59pm on Sep 16, 2024** |
| Grace Due Date: | **11:59pm on Sep 19, 2024** |

Grading Notes: 1) Assignment is due on the Due Date. 2) For each day late after the Due Date, there will be 10% penalty on the assignment's grades. 3) Submission is not accepted after the Grace Date. In other words, you will receive zero pts if your submission is not received by the Grace Date.

Grading: Coding Style 5 pts, Test Cases 95 pts, Total 100 pts.

1. The Triangle Sum Challenge.

$T_n$, the $n^{th}$ triangle number, is the number of points required to construct a triangle with n dots on each side (see Figure 1). In particular, the sum of integers from 1 to n is as follows.

$$T_n = \sum_{i=1}^{n} i$$

Write a recursive function int triangleNum(int n) that returns $T_n$.

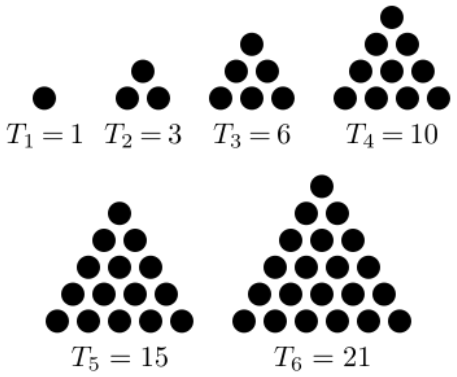


Figure 1: The Triangle Challenge

(https://commons.wikimedia.org/wiki/File:First_six_triangular_numbers.svg)

2. The Fibonacci Challenge.

The Fibonacci numbers are defined by the following recursive formula:

*fib(0) = 0*

*fib(1) = 1*

*fib(n) = fib(n − 1) + fib(n − 2)*

Write a recursive function int rFib(int n) that computes and returns the nth Fibonacci number.

3. Recursive Vector Sum.

Write a recursive function int rSum(const vector<int> &v) that returns the sum of elements in a vector. A helper function might be useful here.

4. Recursive Vector Max

Write a recursive function int rMax(const vector<int> &v) that returns the maximum value in a vector. If the vector is empty, the function should return INT_MIN. Again, consider using a helper function.

5. Recursive Sorting Check

Write a recursive function bool isSorted(const vector<int> &v, int start, int end) that determines whether or not the the elements of v from index start through index end (in- clusive) are sorted in ascending order.

6. The Palindrome Challenge

Write a recursive function bool isPalindrome(string s) that returns true if s is a palindrome and false otherwise. Recall that a palindrome is a string that is the same forwards and backwards. For instance, “racecar” is a palindrome. Once again, a helper function might make things easier.

7. The Binary Search Challenge

Write a recursive function int rBinarySearch(const vector<int> &v, int low, int high, int target) that uses a binary search to find the index of target in the sorted vector v. If target is not in the vector, the function should return -1. You can assume that v will not contain duplicate values.

8. The Subarray Sum Challenge

Given a vector of integers v, a start index start, and a target value target, write a recursive function bool rSubsetSum(const vector<int> &v, int start, int target) to determine whether or not there is a subset of the values in v which add up exactly to target.

For example,

```
vector<int> v = {-5, 13, 19, -27, 8};

int start = 0;

int target = 22;

rSubsetSum(v, start, target);
```

should return true since 22 = (-5) + 19 + 8.

On the other hand,

```
int target = 30;

 rSubsetSum(v, start, target);
```

should return false since no subset of v sums to 30.

As a side note, the subset sum problem turns out to be an interesting, and difficult, problem in computer science.