CS 1428
Fall 2019
Gentry Atkinson
Lab 12

Introduction:

You should be very familiar with functions by now. We have studied calling and defining them over the last few weeks. But let's take a minute to consider what could happen if a function calls itself (or another copy of itself). The intuitive answer is that this would cause an infinite loop of sorts. But this isn't the case if we are careful when crafting our functions. This technique is called recursion.

Every recursive function must have a "**Base Case**", which is some condition which will cause it to return a fixed value. Every call to a recursive function creates a chain of function calls which ends with the Base Case. This final function call then passes a value back up to the function instance which called it. That value is then processed and passed farther up the chain to the next instance of the function. This process continues until every instance of the function has received and returned a value. The first instance terminates last and returns its value to whatever entity called it. Be very careful when creating a recursive function to always include a base case which you know will be reached.

Recursion can be very useful and powerful or very expensive and counter-intuitive. Learning when to use recursion and when to use iteration is part of the process of learning to code.

The purpose of today's lab is to introduce you to recursion.

1- Download the starter file yourName_lab12.cpp from the TRACS site. Give it an appropriate name.

2- Fill in the standard header with your name.

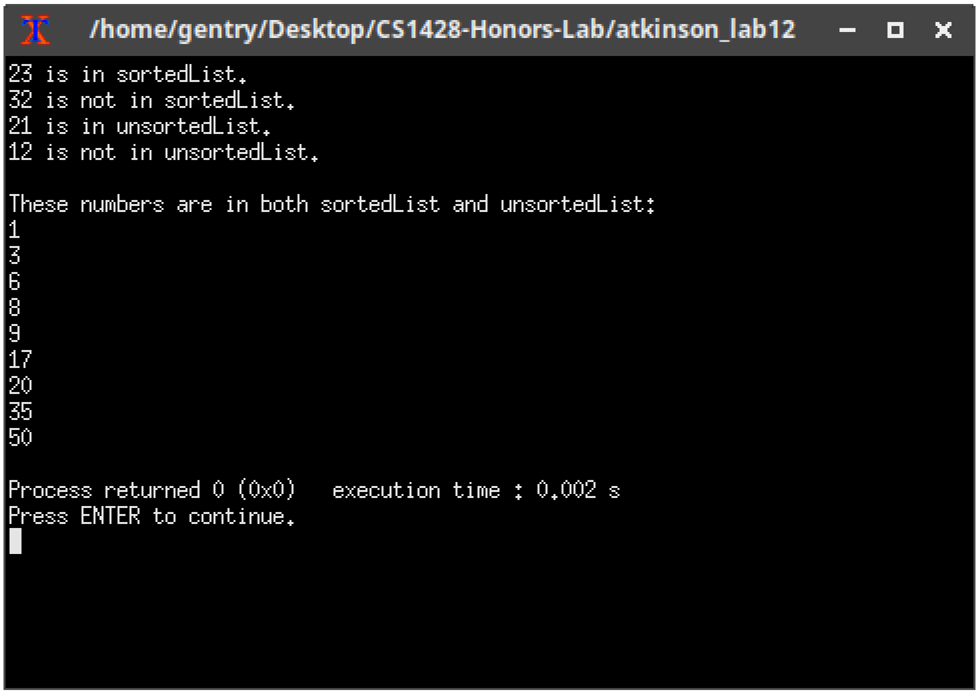
3- You have been given partial implementations for two search functions: Binary Search and Sequential Search. Follow the instructions in the comments to fill in two working functions.

4- Add four function calls to your main function. Use **the best function** for each call.

1. Search sortedList for the number 23 and print the return with a **cout** statement.
2. Search sortedList for the number 32 and print the return with a **cout** statement.
3. Search unsortedList for the number 21 and print the return with a **cout** statement.
4. Search unsortedList for the number 12 and print the return with a **cout** statement.

5- Add a **for loop** to you main function which iterates an **index i** from 0 to 50 (including 50). The body of this loop should use your two search function to print every value of i that is in sortedList and unsortedList.

6- Build and run your code. Correct any errors. Your output should look something like this:



```
23 is in sortedList.
32 is not in sortedList.
21 is in unsortedList.
12 is not in unsortedList.

These numbers are in both sortedList and unsortedList:
1
3
6
8
9
17
20
35
50

Process returned 0 (0x0)   execution time : 0.002 s
Press ENTER to continue.
```

7- Submit your .cpp file to TRACS. You can leave when you're done.