Computer Science 115
Assignment 3
Due March 6th at 11:59pm

Objectives

- Understand that there are different algorithms for accomplishing the same goals
- Understand that some algorithms are much more efficient than others
- Practice writing recursive functions
- Understand the big O running time of binary search on a sorted list

Introduction

In this assignment you will do some experiments with two types of searches: linear search and binary search.

You've been provided with iterative implementations of binary search and linear search. In part two of the assignment you will rewrite these searches recursively.

Part I – Running Time

Study the code for `linearSearch` and `binarySearch` in `IntList.java`.

Modify the code for `linearSearch` and `binarySearch` so that they count the number of iterations the code makes to search for a particular value.

I suggest you initialize a counter variable at the start of the function, increment its value inside the loop and print the result before you return from the function.

Modify the main method in `ListTester.java` (or write your own test program) to complete the table on the last page of the assignment.

NOTE: Depending on the configuration of your java installation, some of the values that you have to complete in the table cannot be computed by running the program. You will need to make educated guesses based on the values that you can compute using the program.

There is no electronic submission for Part I. Hand in your completed table to the drop box on the 2nd floor of ECS.

Part II – Recursive solutions

A recursive function is one that calls itself. In order to be certain that your recursive functions terminate (do not loop forever) you must:

- have at least one base case: a part of the problem be solvable without recursion.
- for each recursive call, make progress towards the base case

Consider the problem of calculating the sum of the integers from 1 to N:

```
int seriesSum ( int n )
{
     if ( n == 1 )
          return 1;
     else
          return n + seriesSum(n – 1);
}
```

The base case here is n == 1. When n is 1, there is no recursive call, the function simply returns 1.

Otherwise, the result of summing the integers from 1 to n is n + the sum of the integers from 1 to n-1.

You should be certain you understand how the code shown above works before move on the next part of the assignment.

Change the implementation of `binarySearch` and `linearSearch` so that both methods use recursion.

Note that while it is very common to see binary search written recursively, writing linear search recursively is just practice using recursion.

You will need to add additional parameter(s) to the search methods to get them to work recursively.

Submission

Submit your completed table in the drop box on the 2nd floor of ECS.

Submit your `IntList.java` that uses recursion via the electronic submission web page.

Part I – Submit on paper

Student Number: _____

List	*Key to search*	*Linear Search*	*Binary Search*
{0,1, 2, 3, ..., 19}	22		
{0,1, 2, 3, ..., 19}	9		
{0,1,2,3,4,..., 1999}	2070		
{0,1,2,3,4,..., 1999}	121		
{0,1,2,3,4,..., 199999}	222222		
{0,1,2,3,4,..., 199999}	38500		
{0,1,2,3,4,..., 99999999}	100000001		
{0,1,2,3,4,..., 99999999}	6320121		

Using the information you gained from testing your program, can you estimate the number of times the loops would be executed searching for the key 2^{256}+1in the list:

{1,2,3,...., 2^{256}}

Provide some justification for your answer. Be sure to answer for linear search and binary search.

Use the back of the page if required.