CSci 115 Spring 2016
Lab 3
(due Tue Feb 23, 6:00pm)

This lab has two parts. In the first part, you will, like you did in Lab 2, complete some exercises from the book. In the second part, you will implement particular linear and binary search algorithms, run some "timing" experiments that compare them on random inputs, and report the results.

Part I

Solve the following exercises from Sections 3.13 and 4.6 of the PDF textbook, and organize them as a PDF file (either scanning nice handwriting, or exporting a word-processing file):

3.1, 3.2; 4.1, 4.2, 4.3, 4.4; 4.6, 4.7

Part II

This is a variation on Exercise 3.7 of Section 3.13 and Project 3.2 of Section 3.14 of the PDF textbook.

Write a program that queries the user for a single positive integer, N, then

- allocates an array of size N (with indices going from 0 to N-1)
- fills this array with integers in order from index 0 to index N-1 in the following way:
  1. start with M = 0
  2. chose a random integer, K, between 1 and 10
  3. store K copies of M in the next K slots of the array
  4. increment M by 1 and continue from step 2 until the array is full
- run a series of 100 random linear and binary searches over this array, keeping track of the number of comparisons done in each case, and report the average of each
- construct a table of the results you get for N = 10, 100, 1000, 10000, 100000, and 1000000 (i.e., 10^n for n = 1,...,6) and graph the results

Additional requirements:

- Use an appropriate random number generator library to choose the random numbers required in this problem. Make sure you properly seed the generator so that it produces different results each time it is run.
- To do a random search in the array, choose a random index from 0 to N-1, look up the value at this position in the array, and search for this value.
- When the value that you are searching for appears multiple times in the array (as it will 9/10 of the time), your linear and binary search algorithms should find the **least** index at which it occurs.
- Annotate both your linear search and binary search code in comments with correct (loop) invariants. The invariants you write for the body of the loop should be (1) true the first time the loop is entered, (2) true each time the loop is re-entered, and (3) imply, when the loop is exited, that the index found in the loop actually contains the value being searched for.

Turning in your lab

Clear out your `turn-in` directory. Then put into that directory, before the 6:00pm deadline on February 23, the following files:

- a PDF file called `part1.pdf` containing your solutions to Part I,
- a program called `part2.cpp` that reads in N and prints out the two averages, and
- a PDF file called `part2.pdf` containing (handwritten or typeset) versions of the table and graph from Part 2.