ICS 311, Spring 2016, Problem Set 01, Topic 2

Due by midnight Tuesday 1/26

#1. Peer Credit Assignment

1 Point Extra Credit for replying

Please list the names of the other members of your peer group for class 1/13 and 1/20 and the number of points you think they deserve for their participation in group work on the two Wednesdays combined.

- You have a total of 6 points to allocate across all of your peers.
- You can distribute the points equally, give them all to one person, or do something in between.
- You need not allocate all the points available to you.
- You cannot allocate any points to yourself! Points allocated to yourself will not be recorded.

#2. Correctness of Linear Search

6 points

(a) Show the pseudocode for Linear Search that you will be analyzing. (It should be code that you understand and believe is correct, so you may revise your group's solution if you wish.) Give each line a number for reference in your analysis.

(b) Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties (page 19).

(Hint: The loop can exit for two reasons. Rather than trying to write and prove an invariant that covers the two cases, use a simpler variant that deals with only the correctness of the after-exit returned value, as it would be too complex to cover both, and the within-loop returned value is easy to show correct.)
#3. Runtime of BinarySearch

9 points
This problem steps you through a recursion tree analysis of BinarySearch to show that it is $\Theta(\lg n)$ in the worst case. We can use the same code as we used in class:

```
Binary­Search(x, A, min, max)
1   low = min
2   high = max
3   while low <= high
4       mid =\[(low + high) / 2\]
5       if x == A[mid]
6           return mid
7     else if x < A[mid]
8         high = mid - 1
9     else low = mid + 1
10  return "NOT FOUND"
```

(The problem involves mathematical equations and drawing diagrams. It is your choice whether to figure out how to do this in Google Docs & Drawing, to do it in your favorite program, or to do your work on paper and scan or photograph it).

(a) Write the recurrence relation for BinarySearch, using the formula $T(n) = aT(n/b) + D(n) + C(n)$. (We'll assume $T(1) = \text{some constant } c$, and you can use $c$ to represent other constants as well, since we can choose $c$ to be large enough to work as an upper bound everywhere it is used.)

(b) Draw the recursion tree for BinarySearch, in the style shown in podcast 2E and in Figure 2.5. (Don't just copy the example for MergeSort: it will be incorrect. Make use of the recurrence relation you just wrote!)

(c) Using a format similar to the counting argument in Figure 2.5 of the text or of podcast 2E, use the tree to show that BinarySearch is $\Theta(\lg n)$ in the worst case. Specifically,
   1. show what the row totals are,
   2. write an expression for the tree height (justifying it), and
   3. use this information to determine the total computation represented by the tree.
#4. Correctness of BubbleSort

15 points

BubbleSort is a popular but inefficient sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order, and doing so enough times that there can be no more elements out of order.

```plaintext
BubbleSort(A)
1   for i = 1 to A.length - 1
2     for j = A.length downto i + 1
```

Let A’ (an array) denote the output of BubbleSort. In order to prove that BubbleSort is correct, we need to show that

1. A’ is a permutation of A; that is, A’ has the same elements as A.

The first part can be proven by showing that BubbleSort never removes or adds items to the array: it merely swaps them in line 4. In this homework, you will use loop invariants to prove the second part (that the resulting array is in order).

(a) State precisely a loop invariant for the for loop in lines 2-4, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in Chapter 2 of CLRS.

(b) Using the termination condition of the loop invariant proved in part (a), state a loop invariant for the for loop in lines 1-4 that will allow you to prove the inequality A'[1] ≤ A'[2] ≤ A'[3] ... ≤ A'[n-1] ≤ A'[n]. Your proof should use the structure of the loop invariant proof presented in Chapter 2 of CLRS.

(c) What is the worst-case running time of BubbleSort? How does it compare to the running time of InsertionSort?