Problem Set 2

> **Due dates:** Electronic submission of .pdf files of this homework is due on **2/3/2017 before 10:00am** on ecampus, a signed paper copy of the pdf file is due on **2/3/2017** at the beginning of class.

Name: (put your name here)

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework.

Signature: _____

Problem 1 (15 points)**.** Consider the following code to find the second largest element:

```
largest := numbers[0];
secondLargest := null
for i=1 to numbers.length-1 do
    number := numbers[i];
    if number > largest then
        secondLargest := largest;
        largest := number;
    else
        if number > secondLargest then
            secondLargest := number;
        end;
    end;
end;
```

This code was provided by someone in response to a question on stackoverflow. (a) How many comparisons does this code make, assuming that `numbers` contains n elements. (b) Give a small example which shows that this is not optimal.

Solution.

Problem 2 (15 points)**.** Describe an algorithm in pseudocode that finds the 2nd largest element in the least possible number of steps. Explain why your algorithm is correct and why it finds the 2nd largest element in the least possible number of steps.

Solution.

Problem 3. (20 points) Give a $(2n-1)$ lower bound on the number of comparisons needed to merge two sorted lists $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ with $a_1 < a_2 < \cdots < a_n$ and $b_1 < b_2 < \cdots < b_n$. [Hint: Use an adversarial method. Why can't you have in general $2n-2$ or fewer comparisons?]

Solution.

Problem 4. (15 points) Solve Exercise 8.1-4 on page 194 of our textbook.

Solution.

Problem 5. (15 points) Consider the task of searching a sorted array `a[1..n]` for a given element w. Show that any algorithm that accesses the array only via Perl-style three-way comparisons using the <=> operator (where $a <=> b$ determines whether $a < b$, $a = b$, or $a > b$), must take $\Omega(\log n)$ steps.

Solution.

Problem 6. (20 points) Suppose that you are given a sorted list L of n distinct elements

$$x_1 < x_2 < \cdots < x_n.$$

You want to search the list for an element y and return $\perp$ if y is not an element of L, and otherwise return the index i if $y = x_i$. You can only access the list using a search procedure `search(i,j,k,y)` that returns one of the following answers: (i) $y < x_i$, (ii) $y = x_i$, (iii) $x_i < y < x_j$, (iv) $y = x_j$, (v) $x_j < y < x_k$, (vi) $y = x_k$, (vii) $y > x_k$. Here it is assumed that `i<j<k`. Use a decision tree to prove a lower bound on the number of calls to `search` that are required to correctly solve the problem.

Checklist:

☐ Did you add your name?
☐ Did you disclose all resources that you have used?
(This includes all people, books, websites, etc. that you have consulted)
☐ Did you sign that you followed the Aggie honor code?
☐ Did you solve all problems?
☐ Did you submit the pdf file of your homework?
☐ Did you submit (b) a hardcopy of the pdf file in class?