CSE 332 Data Abstractions, Winter 2014
Homework 5
Due: Wednesday, February 19, 2014 at the BEGINNING of lecture. Your work should be readable as well as correct. You should refer to the written homework guidelines on the course website for a reminder about what is acceptable pseudocode. Please write your section at the top of your homework.

Problem 1: Algorithm Analysis
The methods below implement recursive algorithms that return the first index in an unsorted array to hold 17, or -1 if no such index exists.

```java
int first17_a(int[] array, int i) {
    if (i >= array.length)
        return -1;
    if (array[i] == 17)
        return 0;
    if (first17_a(array, i+1) == -1)
        return -1;
    return 1 + first17_a(array, i+1);
}

int first17_b(int[] array, int i) {
    if (i >= array.length)
        return -1;
    if (array[i] == 17)
        return 0;
    int x = first17_b(array, i+1);
    if (x == -1)
        return -1;
    return x + 1;
}
```

(a) What kind of input produces the worst-case running time in an absolute “number of operations” sense, not a big-O sense, for first17_a(arr,0)?

(b) For first17_a, give a recurrence relation, including a base case, describing the worst-case running time, where n is the length of the array (e.g. T(n) =…). You may use whatever constants you wish for constant-time work.

(c) Give a tight asymptotic (“big-Oh”) upper bound for the running time of first17_a(arr,0) given your answer to the previous question. That is, find a closed form for your recurrence relation. Show how you got your answer.

(d) What kind of input produces the worst-case running time in an absolute “number of operations” sense, not a big-O sense, for first17_b(arr,0)?

(e) For first17_b, give a recurrence relation, including a base case, describing the worst-case running time, where n is the length of the array. You may use whatever constants you wish for constant-time work.

(f) Give a tight asymptotic (“big-Oh”) upper bound for the running time of first17_b(arr,0) given your answer to the previous question. That is, find a closed form for your recurrence relation. Show how you got your answer.

(g) Give a tight asymptotic (“big-Omega”) worst-case lower bound for the problem of finding the first 17 in an array (not a specific algorithm). Briefly justify your answer.
Problem 2: Sorting Phone Numbers
The input to this problem consists of a sequence of 7-digit phone numbers written as simple integers (e.g. 5551202 represents the phone number 555-1202). The sequence is provided via an Iterator<Integer> - you do not get an array containing these phone numbers and you cannot go through the iterator more than once. No phone number appears in the input more than once but there is no limit on the size of the input. You may assume that phone numbers will not start with 0, although they may contain zeroes otherwise.

Write precise pseudocode for a method that prints out the phone numbers (as integers) in the list in ascending order. Your solution must not use more than 2MB of memory. (Note: It cannot use any other storage – hard drive, network, etc.) In your pseudocode you may only declare variables and arrays of these unsigned data types (these are not real Java data types): bit(1 bit), byte(8 bits), short(16 bits), int(32 bits), long(64 bits). Explain why your solution is under the 2MB limit.

Problem 3: Graph Representations
Suppose a directed graph has a million nodes, most nodes have only a few edges, but a few nodes have hundreds of thousands of edges:

a) In what way(s) would an adjacency-matrix representation of this graph lead to inefficiencies?

b) In what way(s) would an adjacency-list representation of this graph lead to inefficiencies?

c) Design a representation for this sort of graph that avoids all the inefficiencies in your answers to parts (a) and (b).

Problem 4: Topological Sort
Weiss, problem 9.1 (the problem is the same in the 2nd and 3rd editions of the textbook): “Find a topological ordering for the graph in figure 9.79 (2nd ed.)/9.81 (3rd ed.).” For each step, show the in-degree array and the queue in the table format shown below:

<table>
<thead>
<tr>
<th>In-degree Array:</th>
<th>s</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>t</th>
<th>Queue (Front -&gt;End)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Etc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>