CS234: DATA TYPES AND STRUCTURES (F'01)
Assignment 3.

Assigned: 11/01

Due: 11/29, a hard copy at the beginning of class, no extensions possible

Overview: This assignment considers several problems connected with design, implementation, and analysis of data structures and algorithms.

Notes: Solutions will be graded first on correctness, then on presentation. In particular, in order to achieve any points, progress must be made toward a correct solution. Some of the problems are little bit challenging, and you may not be able to solve all of them right away. The following strategy is therefore recommended:

- Spend adequate time thinking about each problem. The solution is likely not to be immediately obvious.
- If you can't come up with a solution, set it aside and come back to it later, perhaps several times.
- Spend adequate time writing up solutions to the problems you are able to solve. If you are aware that your solution is incomplete, make that clear.
- After you have written a solution, set it aside and come back to it later. Read through it and try to improve on the presentation. If there appear to be weaknesses in your argument try to tighten them. In so doing, you may discover that your solution is incomplete.
- Write the final solution for every problem on a **separate, single-sided** sheet of paper, remember to write your name, student id, the problem number, and the page number (if necessary) on **each submitted page**. Make sure your handwriting is legible.
- Be sure you submit your solutions before the due time.

In particular, your final solutions **must** be clear and concise (each problem can be solved in a page or two). Excessively long solutions will receive fewer marks.

Problem 1 (16pt)

Design a *data structure* that implements the stack ADT operations **push** and **pop** for a stack of *integers* and an additional operation **findMin** that finds the smallest element currently held by the data structure. All the operations must run in $O(1)$ worst case time.

Problem 2 (16pt)

Prove (for all $k > 0$) that if the values $1, 2, \ldots, 2^k - 1$ are inserted into an AVL tree in this order the resulting AVL tree is perfectly balanced (Note: a picture is not a proof).

Problem 3 (15pt)

Design an algorithm that *finds* all nodes less than some value, X, in a binary heap. Your algorithm must run in $O(k)$ where k is the number of elements in the heap with value less than X.

Problem 4 (14pt)

Prove that any algorithm that finds an element in a sorted array/list of n elements must take $\Omega(\log n)$ time.

Problem 5 (18pt)

Prove that if a fourth operation, **deleteMin**, that finds and *removes* the smallest element from the data structure is added to operations in *problem 1* then at least one of the operations must take $\Omega(\log n)$ time.

Problem 6 (16pt)

Consider an implementation of AVL tree **insert** operation as described in the text (p.128) that, instead of storing the `height`'s of the subtrees in the nodes of the AVL tree, uses a **height** function defined as

```
private static height(AvlNode t) {
  if (t==null) then return -1
               else return max(height(t.left), height(t.right))+1;
};
```

What is the worst case time of such an insertion routine?

Problem 7 (15pt)

Let $G = (V, E)$ be a (not necessarily binary) *tree* with root s. To this graph add a new node t and edges with "infinite" capacity from each leaf of the original tree to this new node. Design an $O(n)$ algorithm that finds the maximal flow from s to t.

Final Note: Similarly to the midterm, there are up to 110 points possible on this assignment and thus you can earn up to 10 points bonus on this assignment that can substitute for possible shortcomings of A1 and A2.