Computer Science 302 Spring 2013 (Practice) Second Examination, March 14, 2013

Name:____

The entire examination is 345 points.

1. True or False. [5 points each]

   (a) ________ A good programmer should never use linear search.

   (b)

   (c) ________ The height of an AVL tree with n nodes is $O(\log n)$.

   (d) ________ The height of a 2-3 tree with n nodes is $O(\log n)$.

   (e) ________ A *stack* is an example of a *search structure*.

   (f) ________ An item can always be inserted into a min-heap of size n in $O(\log n)$ time.

   (g) ________ A *treap* is an example of a *priority queue*.

2. Fill in the blanks (5 points each blank).

   (a) A 2-3 tree which holds 400 data items must have height at least ________ and at most ________. (Exact answers, please.)

   (b) A condition that is

      i. True for the first iteration of a loop, and

      ii. true at the end of any iteration of that loop, provided it is true at the beginning of that iteration,

   is called a ______________________________ of that loop. [2 word answer]

3. For each of the following blanks, the correct answer is *stack*, *queue*, *list*, *array*, *heap*, or *search structure*.

   (a) *pop* is an operator of _______________ .

   (b) *find* is an operator of _______________ .

   (c) You would use a _______________ to do breadth first search.

   (d) You would use a _______________ to hold the records of the customers of a business.

   (e) You would use an _______________ to keep track of the number of times each word occurs in a Shakespeare play.

(f) You would use a ______________ to match left with right parentheses in an algebraic expression.

(g) Every time you get money, you pay as many bills as possible, in order of urgency. Urgency is determined by due date, not the date that you received the bill.

You would use a ______________ to store your unpaid bills.

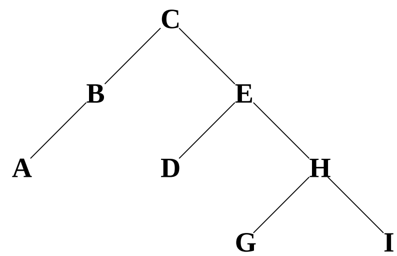
4. [20 points] Build a binary search tree, starting from an empty tree, inserting the following items one at a time: **Moe Abe Joe Nan Ted Kim Sam Ron Dan Sue Zed**. Once an item is inserted into the tree, it is not moved.

5. [20 points] Suppose that the items of a queue are A, H, K, B, T in that order, where A is the front item.

   (a) Sketch the appearance of a circular linked list implementation of that queue.

   (b) Insert the item L into that queue. Show the steps. (You should draw at least two additional figures.)

6. Consider the following AVL tree T.

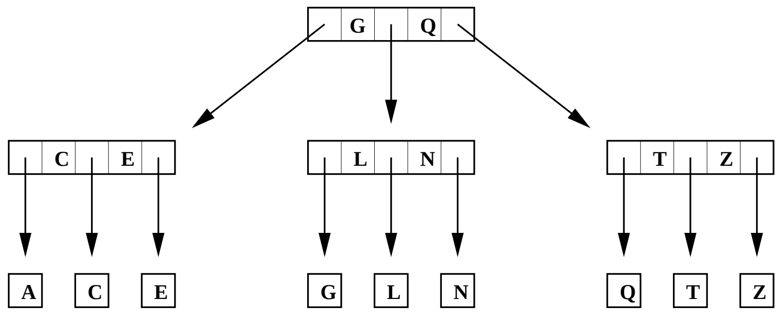


(a) [10 points] An AVL tree should have some additional data, missing in this figure, at each node. Draw in those missing data.

(b) [10 points] List the nodes of T in preorder, inorder, postorder, and level order.

(c) [20 points] Insert **F** into T, and rebalance. Show the steps.

7. [30 points] Consider the 2-3 tree illustrated below. Illustrate the tree after the letter **J** is inserted. (You are not required to show steps, but it doesn't hurt.)

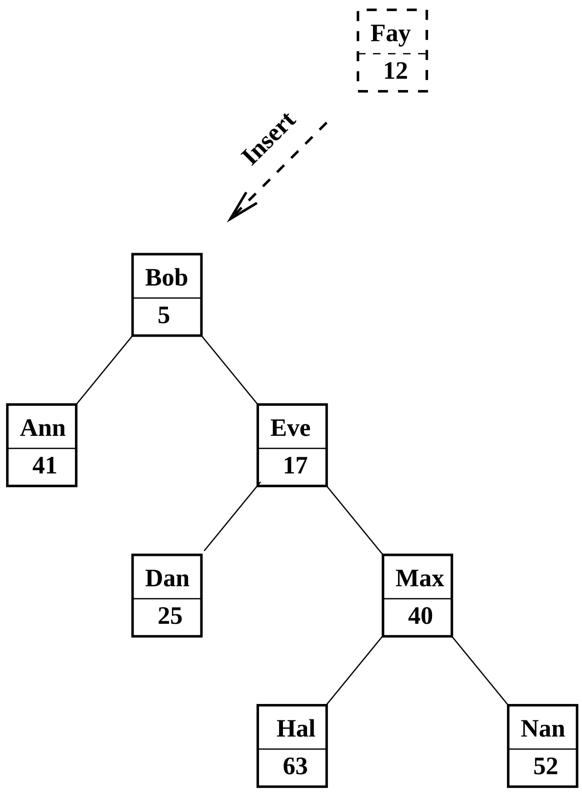


8. [30 points] Consider the heap of 12 elements, implemented as an array as illustrated below. Illustrate the implementation after insertion of the letter **E**. (You are not required to show steps, but it doesn't hurt.)

B	**M**	**K**	**Q**	**N**	**S**	**L**	**Z**	**W**	**P**	**R**	**U**	**X**	**B**	**V**	**W**	**K**	**A**

12

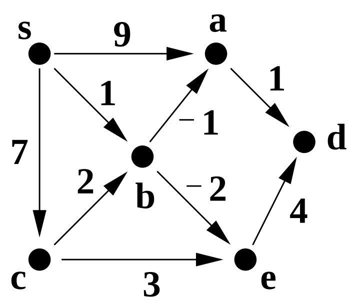
9. [30 points] Consider the treap illustrated below, where the heap key is a randomly chosen integer in the range $0 \ldots 99$. A new item, "Fay," is inserted, and the heap key "12" is chosen. Show the treap after that insertion. Show the intermediate steps.



10. [30 points]

(a) Write a topologicial ordering of the nodes of the weighted directed graph shown in the figure below.

(b) Solve the single source shortest path problem for that graph.



11. [30 points]

By drawing pictures and writing words, define the loop invariant of the partion phase of quicksort.

12. [30 points]

Walk through the steps of heapsort for the following array. Show the complete array at each step where items change position. At each step, indicate the current extent of the heap.

```
Z  B  F  G  P  W  A  R
```