CS140 Final
Fall 2008

Instructions

1. Write your name *clearly* at the top of the exam and on each page of the exam.
2. You must answer all the questions on the exam. The exam has 8 problems, totaling 75 points. Make sure that you have all 8 problems. The remaining 25 points will be determined by your grade on the take-home coding question.
3. Write your answers on the exam pages and hand in the exam when you are finished.
4. You have until the end of the class period to finish the exam. When I call time you must immediately drop your pen/pencil and hand in your exam.
5. Ensure that you write clearly and legibly. Failure to do so may result in the deduction of points.
6. The exam is open note, open book, closed neighbor.
7. Good luck!

---

1. **(14 points)** For each of the following questions choose the best data structure to use from the below list. You may have to use the same answer for more than one question:

Hash Table	Linked List	Array
Binary Search Tree (unbalanced)	AVL Tree	Splay Tree
Stack	Heap	

   a. **Heap:** You are writing a waiting list application for the door person at a popular night club. As each party of customers arrive, the door person assigns the party a priority number based on the party's perceived desirability in the club. The door person then adds them to your computerized wait list and when space becomes available, the door person admits the next party with the lowest priority number. What data structure should you use to implement the wait list?

   b. **Hash Table:** You are writing an address book application. A user should be able to insert (name, address) pairs into the address book, lookup addresses by entering a name, and delete (name,address) pairs from the address book. These are the only operations that your address book must support. What data structure should you use to implement the address book?

   c. **Array:** You are reading input lines from a file. Each input line contains a student name and a midterm grade. With each student you want to

store the set of midterms associated with that student. For each student, the midterms should be stored in the order in which they are read. You know that each student will have 3 midterms. What data structure should you use to store the midterms for each student?

d. **Stack:** When computing taxes on selling the shares of a stock, you are allowed to select a last-in, first-out (LIFO) option, which means that the last shares that you purchased are the first ones that you consider sold. If you have a computer application that reads a set of stock transactions, what data structure should you use to store the stock transactions in order to facilitate implementing the LIFO option?

e. **Binary Search Tree:** You want to read a file that contains the names of contestants on America's Next Top Model and the number of votes that they received. You want to be able to be able to perform a variety of range queries, such as finding the top 5 contestants, the bottom 5 contests, or the number of contestants receiving between x and y votes. What data structure should you use, assuming that you feel the vote totals will be presented in a random order?

f. You are reading the names of golfers and their scores from a file. Each input line contains the name of a golfer, the golf course where the round was played, and the score for that round. For each golfer you want to store the information for each round played by that golfer. There is no limit to the number of rounds per golfer. After reading the input, the user should be able to query for a golfer and obtain the golfer's average score and a print out of the rounds played by that golfer. The insertion operation and the query operation are the only operations performed on the data.

    1. **Hash Table:** What data structure should be used to store the golfers?

    2. **Linked List:** For each golfer, what data structure should be used to store the information about the golfer's rounds? You should assume that each round has one record and the rounds may be stored in any order.

2. **(4 points)** Write a one line statement that divides an integer, n, by 8 and assign the result to n. Your answer **must** use a bit expression, not division.

    n = n >> 3;

3. **(10 points)** Show the hash table that results if the integers 51, 21, 38, 30, 44 are inserted into the following hash table using:
    a. Linear probing, and
    b. the hash function h(k) = k % 7

    0: 21

    1:

    2: 51

    3: 38

    4: 30

    5: 44

    6:

4. **(10 points)** Show the hash table that results if the integers 51, 46, 23, 81, 21, 38, 30, 44, 56, and 69 are inserted into the following hash table using:
    a. separate chaining, and
    b. the hash function h(k) = k % 7

    You should append integers to the end of your lists and you should represent lists as comma separated lists of integers:

    0: 21, 56

    1:

    2: 51, 23, 30, 44

    3: 38

    4: 46, 81

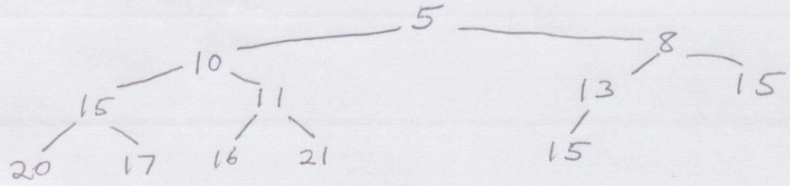
    5:

    6: 69

5. **(12 points)** Suppose you are given the following heap in array format:

5 10 8 15 11 13 15 20 17 16 21 15

a. Draw the heap represented by this array

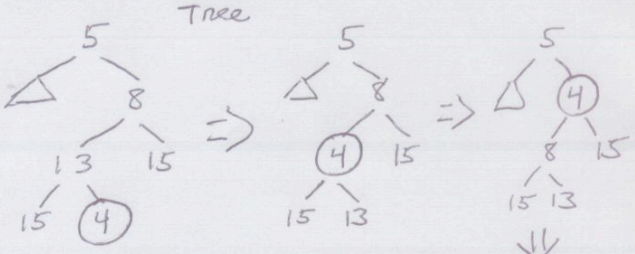


b. Show the heap that results from inserting 4. If you want partial credit then show your intermediate steps. It is ok to either draw the resulting heap or to represent it as an array.

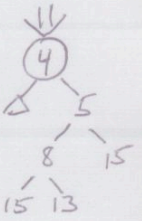
Array

0	1	2	3	4	5	6	7	8	9	10	11	12	13
5	10	8	15	11	13	15	20	17	16	21	15	4	
				4								13	
		4		8								13	
4	5			8								13	
4	10	5	15	11	8	15	20	17	16	21	15	13	

Tree

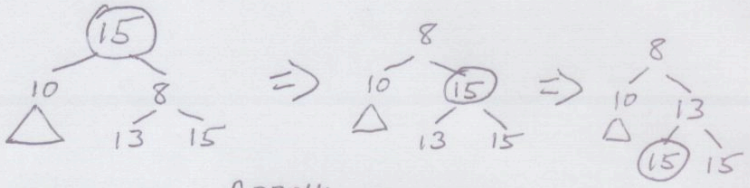


c. Show the heap that results from performing a deleteMin on the *original* heap. If you want partial credit then show your intermediate steps. It is ok to either draw the resulting heap or to represent it as an array.



5 gets deleted – 15 gets promoted and pushed down

Tree

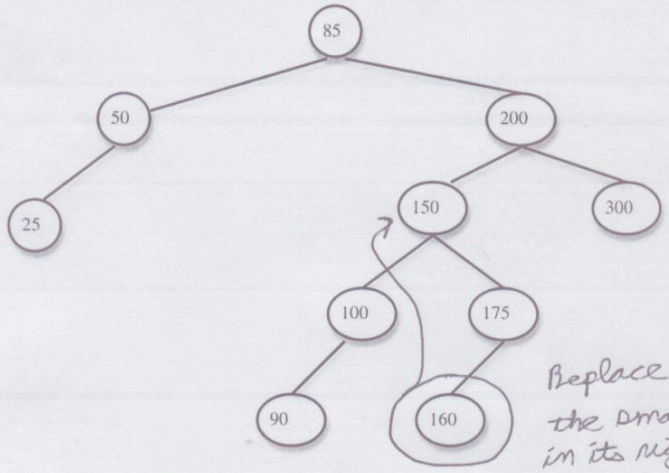


Array

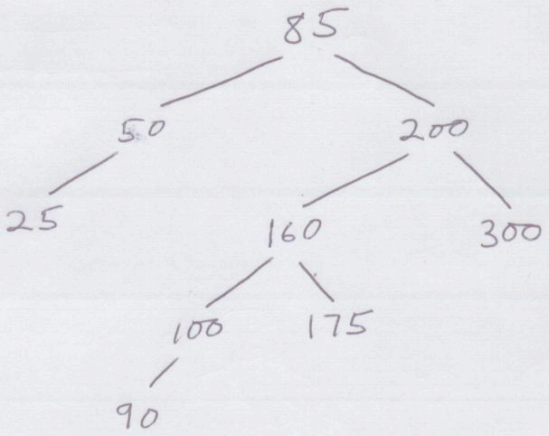
	0	1	2	3	4	5	6	7	8	9	10	11	12
Initial Array:	5	10	8	15	11	13	15	20	17	16	21	15	
Delete 5, promote 15	15	10	8	•	•	•					16	21	
Promote smaller of 10, 8	8	10	15	15	11	13	15	20	17	16	21		
Promote smaller of 13, 15	8	10	13	15	11	15	15	20	17	16	21		

circled elements denote changed elements or children to be considered for next step

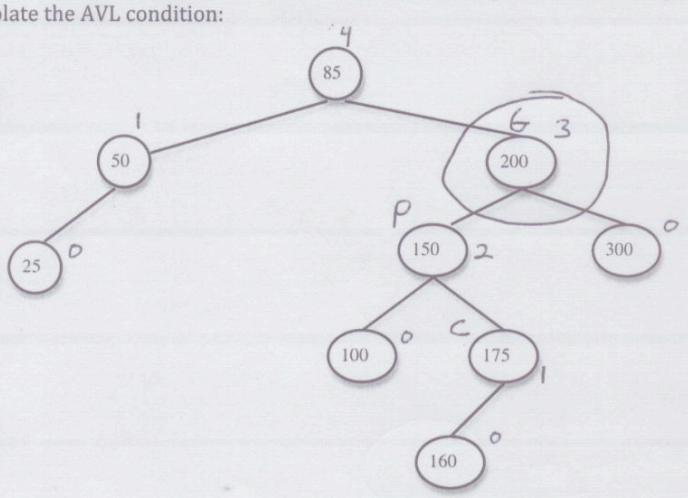
6. **(5 points)** Show the binary search tree that results if 150 is deleted from the tree below:



Replace 150 with the smallest element in its right subtree



7. **(10 points)** 160 has just been inserted into the following AVL tree, causing it to violate the AVL condition:



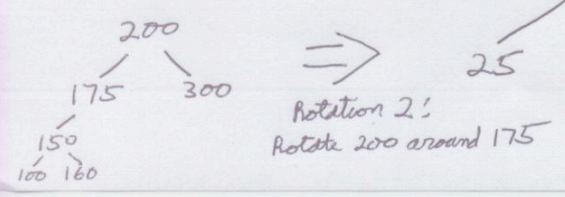
a. Identify the **bottom-most** node that violates the AVL condition and explain why that node violates the AVL condition.

200 – It is the first node on the path from the inserted node, 160, to the root that has a height imbalance of 2 between its left and right subtrees

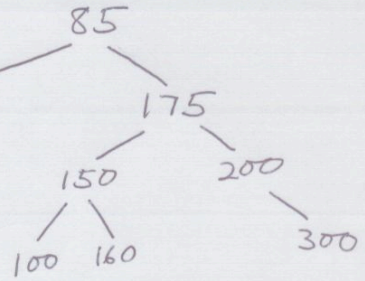
b. Use the proper rotation(s) to rebalance the above tree so that it becomes a legitimate AVL tree.

A double rotation with 175 as the child, 150 as the parent, and 200 as the grandparent is required. We determine what type of rotation is needed by examining the last two links on the path from 160 to 200. Since the last two links are in opposite directions, a double rotation is needed:

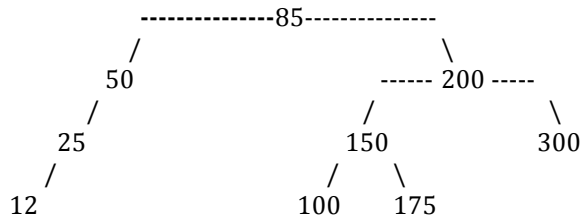
Rotation 1: 150 rotated about 175



Rotation 2: Rotate 200 around 175



8. **(10 points)** 12 has just been inserted into the following AVL tree, causing it to violate the AVL condition:

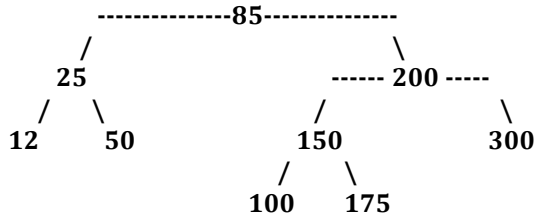


a. Identify the **bottom-most** node that violates the AVL condition and explain why that node violates the AVL condition.

The bottom-most node that violates the AVL condition is 50. The height of 12 is 0, and hence the height of 25 is 1. 50 has an empty right subtree, whose height is -1. Hence there is a difference of 2 between 50's left subtree, rooted at 25, and its right subtree, which is empty.

b. Use the proper rotation(s) to rebalance the above tree so that it becomes a legitimate AVL tree.

This imbalance is a left-left imbalance, and hence can be fixed with a single rotation about 25. 50 will rotate down to become the right child of 25 and 25 will float up to become the left child of 85:



9. **(25 points)** Declare and write a function named **count_nodes** that counts and returns the number of nodes in a binary search tree. For example, if your function were applied to the tree in question 6, it would return 10. Assume that you have the following struct for a binary tree node:

```
typedef struct bnode {
        int key;
        struct bnode *left_child;
        struct bnode *right_child;
} BNode;
```

Further assume that your function takes a single argument, which is a pointer to the tree's root node.

In order to count the number of nodes in the tree, you need to 1) count the number of nodes in your left subtree, 2) count the number of nodes in your right subtree, 3) add the two counts together and add 1 for yourself. For example, in the tree in question 6, there are 2 nodes in the root's left subtree and 7 nodes in the root's right subtree. Adding the two counts together and adding 1 for the root produces an answer of 10. This corresponds to a post-order traversal of tree, where you first process your left subtree, then your right subtree, and finally yourself. One can write an equation for the number of nodes in a subtree as follows:

Number of Nodes(empty tree) = 0
Number of Nodes(tree rooted at n) = Number of Nodes(n->left_child)
+ Number of Nodes(n->right_child) + 1

This equation can be implemented in our post-order traversal as follows:

```
int count_nodes(BNode *node) {
    if (node == NULL)  // empty tree
        return 0;
    else
        return count_nodes(node->left_child)
            + count_nodes(node->right_child) + 1;
}
```