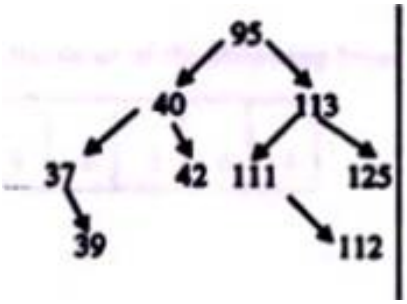
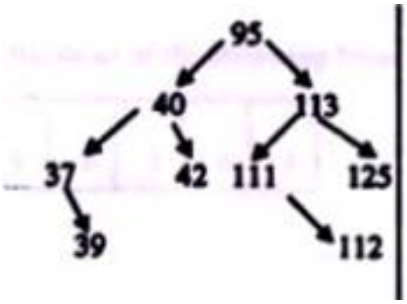
CS61B Midterm 2 Fall 1997

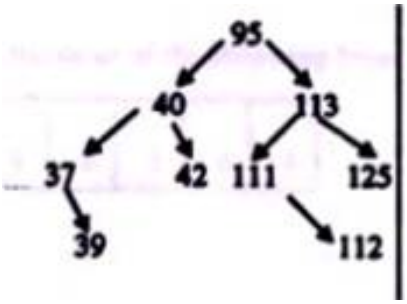
1) (2 points) Given the following BST, show its value after deleting 95.



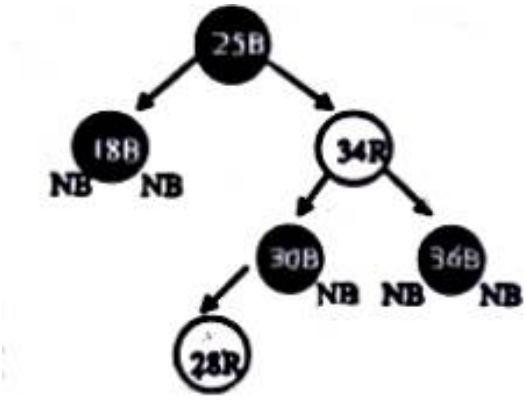
2) (2 points) Given a BST, there are some keys that, if inserted will increase the height of the tree. For the BST below, list all the integer keys (distinct from those already in the tree) for which this is true. If no such keys exist, explain why not. (Each key should increase the height if inserted alone, not with other keys.)



3) (4 points) For a given set of keys there are many possible BSTs. Right and left rotations (as defined for Red-Black trees), produce legal BSTs from others. For the BST given below, give another BST with the same keys that CANNOT be formed by performing a sequence of left and right rotations on T. If no such tree exists, explain why not.



4) (2 points) Given the following Red-Black tree, show its value after inserting the key 29. (Mark the red nodes with an "R" and black nodes with "B" and show the black nulls at the leaves as in your homework assignment.



5) (2 points) Show the value of the following heap after performing one remove() operation.

--	15	5	9	2	3	6	4

Fill in answer above (leaving unused elements blank).

6) (4 points) In class we discussed the array representation of a complete binary tree. This same idea can be used for storing complete k-ary trees in which each node has at most k children, rather than 2 children. Assume the first element of the array is left blank, as in class.

a) Give a formula for finding the jth child of node i in such a tree.

b) Give a formula for finding the parent of node i in such a tree.

7) (2 points) As part of a hash table implementation that uses open address hashing, someone suggests a probe function p (x) = x*x. What is wrong with this as a probe function? (1 sentence limit.)

8) (6 points) Someone has asked you to design a data structure to hold objects with two keys, one key (k1) is an int and the other (k2) is a String. There are 5 important operations:

insert(Object) -- takes an object containing both keys;

lookupByKey1(int) -- takes only an integer key to lookup by k1

lookupByKey2(String) -- takes only a String key to lookup by k2

deleteByKey1(int) -- delete by k1

deleteByKey2(String) -- which takes a String key.

All of these operations should work in O(1) time. Give a 1 sentence description of your data structure and draw a picture of it. Your picture should contain at least one stored object with fields k1 and k2.

9) (6 points) Fill in the function below to perform a right rotation on the left child of a tree node p. You may assume that p, its left child, and its left child's children are all non-null.

class TreeNode { left TreeNode; right TreeNode; Object data }

void rotateRightOnLeftChild (TreeNode p) {

10) (3 points) The performance of quickSort depends on the order of elements in the input array. Give an input array

containing the integers 0 through 8 that will make quickSort perform at its worst, i.e., will maximize the number of comparisons requred by quickSort. (Assume the pivot is chosen as the middle element as in the book and notes.)