CSE 373 Spring 2006

Data Structures and Algorithms

Midterm

Wednesday May 17th

NAME : _____

Do all your work on these pages. Do not add any pages. Use back pages if necessary. Show your work to get partial credit.

This exam is worth 50 points. After each question, you will find the number of points it is worth. You should spend approximately x minutes on a question worth x points.

1. 10 points
2. 4 points
3. 10 points
4. 9 points
5. 8 points
6. 9 points

1. (10 points) AVL trees This question continues on the next two pages.

(a) (3 points)
Give the definition of an AVL tree. Could you define a BST that is "more balanced" than AVL trees. If so, give the definition. If not, why not?

An AVL tree is a BST such that for each node the absolute value of the difference in heights of its children is 0 or 1.

A more balanced BST would be a completely balanced BST where all levels are full except possibly the last one.

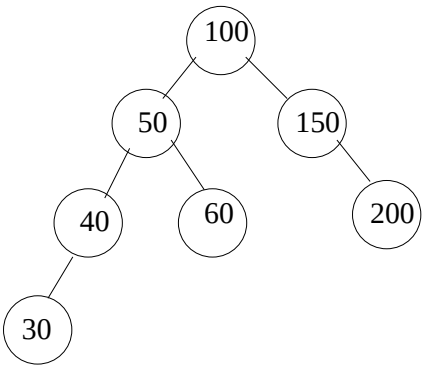
(b) (1 point)

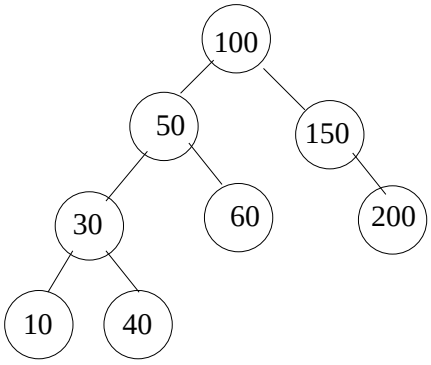
Answer True or False:

A nice property of AVL trees is that each of *Find*, *Insert* and *Delete* takes O(logn) time.

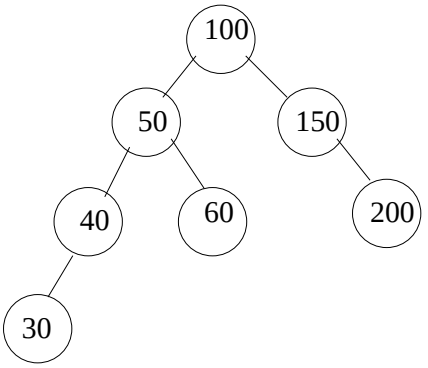
True

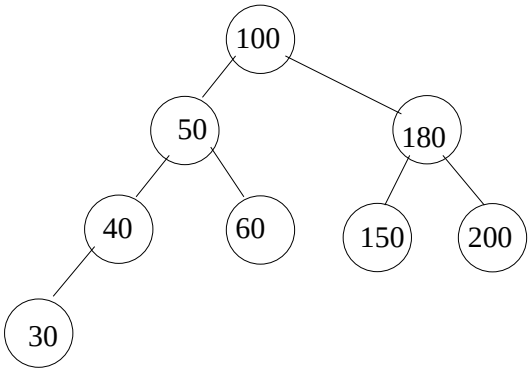
(c) (2 points)
Given the AVL tree below, show the AVL tree that would result after inserting the key of value 10.



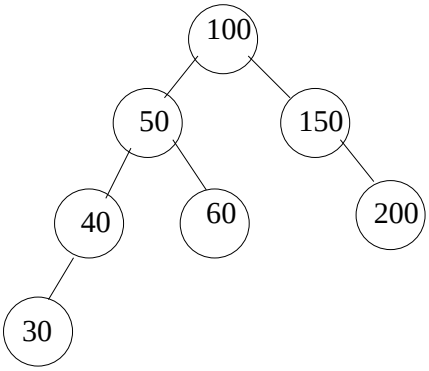


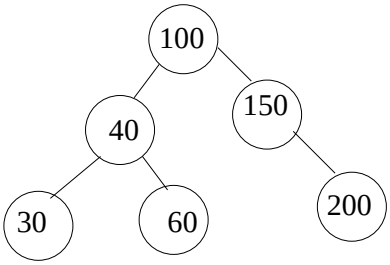
(d) (2 points)
Given the AVL tree below, show the AVL tree that would result after inserting the key of value 180.





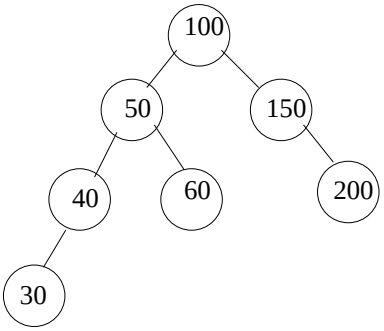
(e) (2 points)
Given the AVL tree below, show the AVL tree that would result after deleting the key of value 50.

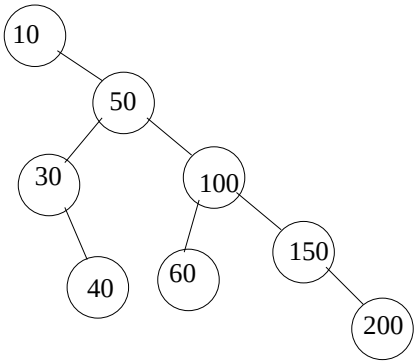




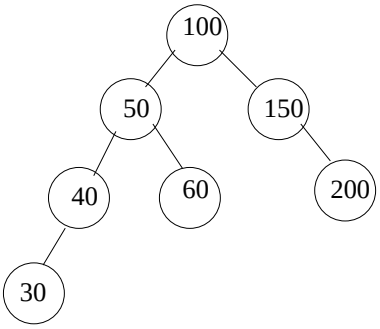
2. (4 points) Splay trees

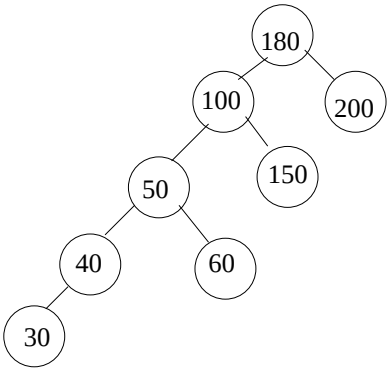
(a) (2 points)
Given the BST below, show the BST that would result after inserting the key of value 10 if *splaying* is performed starting at the node that was inserted.





(b) (2 points)
Given the BST below, show the BST that would result after inserting the key of value 180 if *splaying* is performed starting at the node that was inserted.





3. (10 points) Splay trees. This question continues on the next page.

(a) (1 point)

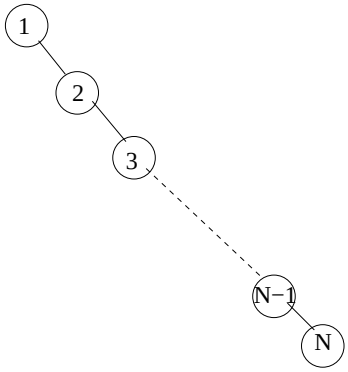
Answer True or False:

A nice property of splay trees is that each of *Find*, *Insert* and *Delete* takes O(logn) time.

False

(b) (3 points)
The keys of value N, N-1, N-2,.., 4, 3, 2, 1 are inserted in this order in a splay tree. What is the final configuration of the tree? What is the cost in Big-Oh notation of each insert operation.

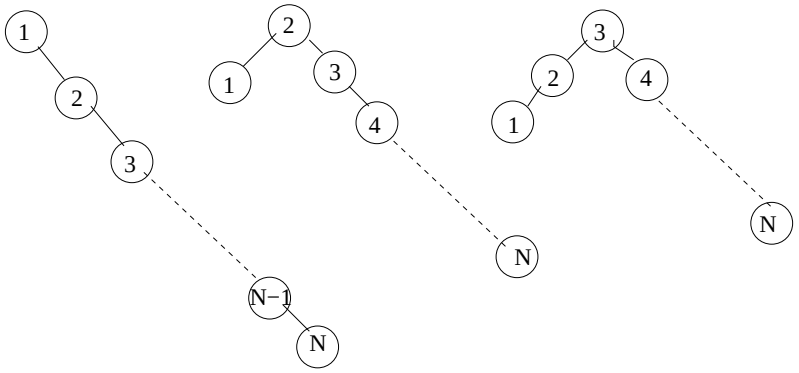
Each insert takes O(1)



(c) (4 points)
Assume now that the next 100 operations will be a mix of only *Find(1), Find(2)* and *Find(3)*, i.e., search in the tree for either the key of value 1, or the key of value 2, or the key of value 3. After each successful search, splaying is performed from the node where the key was found.

What are the 3 tree configurations that are possible after these 100 operations are completed?

What is the cost in Big-Oh notation of each *Find* operation?



Each operation takes O(1)

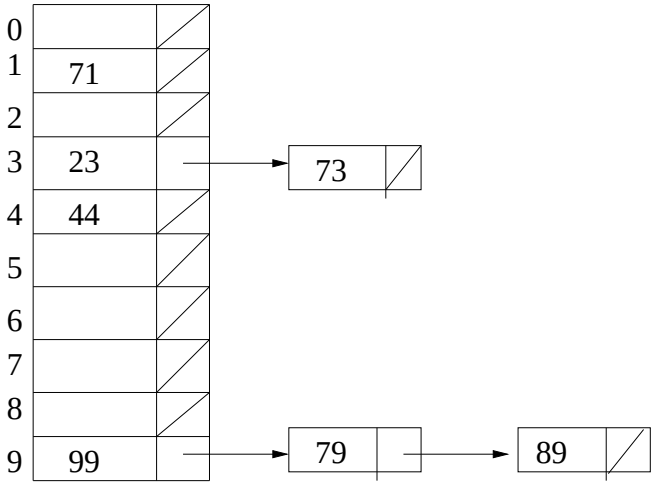
(d) (2 points)
The next operation is *Find(N)*. In terms of Big-Oh notation, is any of the 3 configurations faster than the others? If so, what is its Big-Oh time complexity? If not what is the Big-Oh time complexity of *Find(N)* in the three configurations?

The 3 configurations yield the same Big-Oh complexity of O(N)

4. (9 points) Hashing. This question continues on the next page.

Given a hashing table of 10 entries, the hash function $h(x) = x \ mod(10)$ and the input sequence {71, 23, 73, 99, 44, 79, 89}

(a) (3 points)
Show the resulting hash table when collisions are resolved with separate chaining.



(b) (2 points)
Show the resulting hash table when collisions are resolved with open addressing and linear probing.

0	79
1	71
2	89
3	23
4	73
5	44
6	
7	
8	
9	99

(c) (1 point)
For case (b) what is the load factor of the table?

load factor = 7/10 = 0.7

(d) (2 points)
Give one advantage and one drawback of linear probing over quadratic probing.

Advantage of linear probing: will always find a slot in the table if there are still some slots left (quadratic probing might not).

Quadratic probing might reduce the number of collisions by avoiding primary clustering

(e) (1 point)

Answer True or False:

Hashing with open addressing works much better when the capacity of the table is NOT a prime number

False

5. (8 points) Sorting

(a) (2 points)
Assuming a random distribution of keys what are the time and space requirements (in Big-Oh notation) of Mergesort?

Time: O(nlogn)

Space: O(n)

(b) (2 points)
Assuming a random distribution of keys what are the time and space requirements (in Big-Oh notation) of Quicksort?

Time: O(nlogn)

Space: O(logn). Although Quicksort sorts "in-place" extra storage is needed for the recursive calls, or keeping on a stack the left (right) and middle indices of the non-yet sorted partitions in the iterative method.

(c) (1 point)
If you knew that the keys are already almost sorted, what sorting algorithm would you *use* among Insertion Sort, Heapsort, Mergesort, and Quicksort?

Insertion Sort

(d) (1 point)
If you knew that the keys are already almost sorted, what sorting algorithm would you *avoid* among Insertion Sort, Heapsort, Mergesort, and Quicksort?

Quicksort

(e) (2 points)

Which of Insertion Sort, Mergesort, and Quicksort is not stable?

Quicksort

6. (9 points) Data structure design . This question continues on the next page.

Joe, Bob and Sue are employees of the same corporation. The inventory of the products they sell is a database of over 1,000 entries where each entry has:

- A catalog number which is a string of 6 characters: 3 letters followed by 3 digits
- The number of products in stock for that catalog number: an integer
- The cost of the product: a real number
- A description of the product: a variable length string of at most 20 characters

(Note: the questions to follow do not depend on the cost and the description).

Joe, Bob and Sue discuss a number of data structures that could be best applied to their individual needs. The choices are:

Arrays	Linked Lists	Binary Search Trees
Heaps	AVL Trees	Splay Trees
Hash Tables		

Joe is a salesperson. His job is to convince the customer to buy a product. Once the customer has chosen a product by looking at the printed catalog, the only thing he needs is to find its availability and decrease the number in stock (if available). Give his choice for a data structure (choose from one of the above) which will lead to the most efficient way to handle his task and explain his reason for the choice in one or two sentences:

The main task is "Find(product)" so a hash table with the product number as a key is the most appropriate structure. Joe will perform his task in O(1)

Bob handles emergency situations. As soon as a product has its stock below a certain threshold he should reorder some from the manufacturer. Give his choice for a data structure (choose from one of the above) which will lead to the most efficient way to handle his task and explain his reason for the choice in one or two sentences:

The main task is "Findmin(availability)" which is best handled with a binary heap (priority queue) with the minimum property based on stock availability. Bob can do a Findmin in O(1) and see if the product at the root of the heap needs reordering.

Sue has more of a managerial position. Her job is to look at the overall picture. She has already noted that 82% of the orders are for 17% of the products. She would like to periodically peek at a list ordered by product number of recently purchased items. Give her choice for a data structure (choose from one of the above) which will lead to the most efficient way to handle her task and explain her reason for the choice in one or two sentences:

The main task is to list the most recently used items. The 82-17% ratio argues strongly for a splay tree. By traversing (part of) the tree in inorder, an O(number of items she wants to look at) process is achieved.