CS 315 Data Structures Practice Questions for mid-semester # 2

(Date of the test: November 13, duration: 75 minutes)

1) Exercises 5.1, 5.2

3) Write a member function for the heap class that takes as input an index j and updates the heap by removing the key stored in index j. (Assume j <= currentSize.) The function should return the deleted key.

4) Consider an open hashing table. Suppose the hash table currently contains the keys 23, 8, 41, 33, 34, 19, 12. Assume also that U = {1, 2, … , 40} and that the hash function used is h(x) = x mod 11. (a) If one of the current keys in the table is being searched with uniform probability, what is the expected number of comparisons performed? (b) if one of the keys currently not in the table is being inserted into the table (again from uniform distribution), what is the expected number of comparisons performed? (Note that the insertion algorithm searches the list h(x) to the end before inserting the key x.)

5) Answer (3) assuming that closed hashing is used. Assume that the rehashing strategy used is linear probing with increment d = 3.

6) If the LPT scheduling algorithm is implemented using an unsorted array (instead of a heap) what will be the time complexity? (Express your answer as a function of n = the number of jobs and m = the number of machines).

7) Given an array of n keys, find the maximum k<= n such that the first k keys of the array form a min-heap. For example, if the array contains keys 1 3 5 7 6 2 11 4 8, the output is 5.

8) Define the following terms: (a) heap property (b) complete binary tree (c) full binary tree.

9) Exhibit a max-heap with 12 nodes in the form of a binary tree.

10) What is the result of inserting 12 into the heap of Figure 6.5 (a), page 217? What is the result of performing DeleteMin on the resulting heap?

11) Let A be an array of integers in which some keys are stored in indices 1 to k. Write a procedure that takes as input A and k, and determines if A[1 : k] forms a min-heap. What is the time complexity of this algorithm?
Hint: Check at each node (except the root) that its key is >= its parent key.

12) How many different min-heaps can you form using the keys 1, 2, 3, 4, 5, 6 and 7? Hint: The root must contain 1, the remaining 6 keys can be arbitrarily split into two groups A and B each with 3 numbers and can be used to fill the left and the right subtree of the root.

13) What is the smallest (largest) number of nodes in a heap of height 6? What is the height of a heap with 200 nodes?

14) Exercise 6.1, 6.2, 6.3

15) Here is a possible solution to Problem 3 (which is to remove a key at position j of a heap): move the key at A[currentSize] to A[j] and call *percolateDown* (j). Will this work? If so, justify it. Otherwise, give an example for which it fails.

16) Write a procedure to delete (and return) the second smallest key from a min heap. Your procedure should perform only a constant number of additional operations besides calling INSERT or DELETEMIN. What is the complexity of your procedure.

17) Given the following list of jobs (the list contains their processing times), display how the jobs are assigned to machines by the LPT algorithm. (LPT algorithm is not described in the text, but can be found in the class notes.) Recall that the algorithm first sorts the jobs in decreasing order of their lengths.

(2, 14, 3, 4, 16, 6, 5, 3, 8)

18) Exhibit the binary search tree that results by inserting the following sequence of keys: 8, 11, 3, 12, 35, 8, 14, 23, 1.

19) What is the height of the above tree? What is the depth of the node containing the key 14? Recall that the height of the binary search tree is defined as the number of edges in the longest path from root to a leaf.

20) There are 24 possible ways in which we could insert the keys 1, 2, 3 and 4 into a (initially empty) binary search tree. Of these, how many would result in a tree of height = 3? (For example, the sequence 1, 4, 3, 2 will produce such a tree.) Answer the same question for the collection {1, 2, 3} and {1, 2}. Can you guess a general formula for the number of input orders that result in a binary search tree of height N – 1 when you are inserting N distinct keys ?

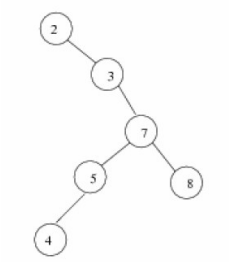
21) Exhibit the expression tree corresponding to the expression: ((a + b * c))* d + e * (f + g)

22) Preorder traversal of a binary search tree produces the sequence 5 3 18 13 9 14 21 24. Construct the tree.

23) Given a pointer to a node N in a binary search tree and a key x, you are to write a procedure that finds outputs the node in the subtree rooted at N that contains the smallest key larger than x. If such a node does not exist, it should a NULL pointer.

HINT: compare x with key at N. If key(N) < x, continue the search recursively on the right subtree. If key(N) = x, then output the node containing the minimum key on the right subtree. If key(N) > x, then call recursively to find the successor of x in the left subtree of T. If such a node exists, then it must be returned. Else the root node is returned.

Example: Consider the binary search tree shown below. If n is the node containing 7, and x is 4 (or 4.7), the output should be the node containing 5. If x = 8, NULL pointer should be returned.



24) The following recursive procedure takes as input a binary tree node T, and answers true (false) if T represents (does not represent) a binary search tree. Determine if the procedure is correct. If it is correct, prove that it is correct. If not, construct an input for which it fails.

```
boolean checkTree ( TreePtr T)
{
 if ((T == null) || (T -> left == null && T->right == null))
   return true;
 else if (checkTree (T->left) && checkTree(T->right)
        && (T->key > T->left->key) &&  (T->key < T->right->key))
 return true;
 else return false;
}
```