CS 315 Data Structures **Spring 2008**

Mid-semester test # 2 April 24, 2008

(Duration: 75 minutes)

1) Let A be a heap of size 4 with keys A[0] = 3, A[1] = 6, A[2] = 4, A[3] = 11. What is the result of inserting 2 into the heap? Exhibit the result of performing *deleteMin* on the resulting heap? **6 points**

2) Exhibit the binary search tree that results by inserting the following sequence of keys: 18, 1, 3, 42, 35, 8, 14, 23, 1. **6 points**

3) Let H be a closed hash table of size 29 (indices 0 to 28) in which the following set of keys are currently stored: { 34, 3, 4, 23, 20, 21, 15, 74, 97, 9}. What is the successive sequence of indices probed while inserting the key 5 in the case of linear probing with increment d = 5? Assume that the hash function used is h(x) = x mod 29. **6 points**

4) Consider a binary search tree application in which the same key can occur more than once. This is handled in the application by using an additional field **Count** that contains the number of occurrences. Rewrite the procedure **insert** in this case. The standard insert is shown below. (If you need a different constructor, you should write one.) **8 points**

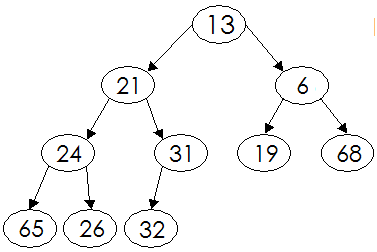
```
BinaryTree* insert(BinaryTree* T, int x) {
// inserts x into T and returns a pointer to the resulting tree
if (T==null) return BinaryTree(x, null, null);
// call to a constructor that creates a single node
else if (T->key < x)
 { T->Right = insert(T->Right, x); return T;}
else if (T->key > x)
 { T->left = insert(T->left, x); return T;}
else return T; // do nothing; x is already in the tree
}
```

5) (a) Exhibit the unique binary search tree whose preorder traversal prints the sequence 41 21 15 38 23 29 25 30.

(b) A binary search tree contains the following keys: 18, 1, 3, 42, 35, 8, 14, 23, 1, 11, 55, 91. If the keys in the tree are printed using inorder traversal, what is the output sequence printed? 8 points **4 + 4 points**

6) Write a procedure *testHeap* that takes as input an array A and an integer *currentSize* that represents the number of keys stored in array A and another integer j, and outputs true (false) if the keys the subtree rooted at node j forms a min-heap. **10 points**

Example: Let A be the array drawn as a complete tree shown below (with currentSize = 10). If *testHeap* is called with j = 2, the output should be ***true***, with j = 1, the output should be ***false***.



Hint: recursive solution is easier than an iterative one.

8) Recall that we presented an algorithm to remove a key at position j (which is also described in the text.) Here is another possible way to remove a key at position j: move the key at A[currentSize] to A[j] and call *percolateDown* (j). Will this work? If so, prove it. Otherwise, give an example for which it fails. **8 points**

9) Write a procedure to delete the second smallest key from a min heap. Your procedure should perform only a constant number of additional operations besides calling INSERT or DELETEMIN. (It should return the second smallest key.) What is the complexity of your procedure? Express your answer using O notation in terms of N = the number of nodes in the heap**.** **8 points**