Homework Assignment 6

1. Given the following binary search trees, show the structure of the tree after a balancing operation has been performed on it. Assume that when selecting the middle, we always round down (or perform the integer division).

2. Given the above trees, show their preorder and postorder traversals.

3. Given the left tree in Problem 1, show the tree after the following operations. Assume that remove operations use the successor to replace a removed node when appropriate.
   (a) add(14)
   (b) add(33)
   (c) add(35)
   (d) remove(39)
   (e) remove(10)
   (f) add(16)
   (g) remove(25)

4. Enter the following nodes into an AVL tree: 45, 12, 3, 98, 20, 30, 75, 55, 100, 0, 60, 85. For each node that triggers a rotation, state what node it is and show the tree after the necessary rotation.

5.
Given the AVL tree above show what the tree looks like after the following nodes are removed: 75, 100, 98.

6. Write two methods that given a node of an AVL tree performs an RR and an RL rotation and returns a reference to the new root of the subtree. You can write psuedocode. Make sure you document your code.

7. Write a method of a binary tree that determines the size of the tree. You can write pseudocode. You cannot assume that there is a data field storing the size of the tree.

8. A priority queue containing characters is implemented using max-heap and stored as an array. The capacity of the array used is 10 elements and the first 7 locations are occupied (indexes 0 through 6). Show the array after each of the following operations is performed (each operation should be modifying the array resulting from the previous step).

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

The name of the priority queue is pq.

(a) pq.enqueue(‘W’);
(b) pq.dequeue();
(c) pq.enqueue(‘C’);
(d) pq.enqueue(‘K’);
(e) pq.dequeue();
(f) pq.dequeue();

HINT: it might be helpful to draw out the tree to decide visually where the nodes are added to and removed from.

9. Given the following node definition for a general tree

```plaintext
2
```

Given the AVL tree above show what the tree looks like after the following nodes are removed: 75, 100, 98.

6. Write two methods that given a node of an AVL tree performs an RR and an RL rotation and returns a reference to the new root of the subtree. You can write psuedocode. Make sure you document your code.

7. Write a method of a binary tree that determines the size of the tree. You can write pseudocode. You cannot assume that there is a data field storing the size of the tree.

8. A priority queue containing characters is implemented using max-heap and stored as an array. The capacity of the array used is 10 elements and the first 7 locations are occupied (indexes 0 through 6). Show the array after each of the following operations is performed (each operation should be modifying the array resulting from the previous step).

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>

The name of the priority queue is pq.

(a) pq.enqueue(‘W’);
(b) pq.dequeue();
(c) pq.enqueue(‘C’);
(d) pq.enqueue(‘K’);
(e) pq.dequeue();
(f) pq.dequeue();

HINT: it might be helpful to draw out the tree to decide visually where the nodes are added to and removed from.

9. Given the following node definition for a general tree

```plaintext
2
```
Node
definition

*int data
*Node.firstChild
*Node.nextSibling

write a method that given a reference to the node prints to the screen the values stored in all of this
node’s children.

10. Convert the following expressions to postfix and to prefix notation

(a) \((3 + y/x) * (x * 2 - 1)\)
(b) \(17 + (15 / 3 - 6) * (-9 + 7*2)\)

11. Given the following min-heap, show the final state of the heap after executing the following opera-
tions:

- add(1),
- add(4),
- remove().

```
3
8 5
15 10 6 9
17 20 12
```

12. Show the array representation of the heap in the previous question.

13. Specify the steps needed to sort a list of last names stored in a text file. You should use heap-sort.
   For each step, specify the complexity (using order notation) of the operations.

14. Specify the characteristic properties and requirements of the following trees:

   (a) binary tree
   (b) binary search tree
15. Specify the worst case complexity of the add/insert and remove (item specified as a parameter) operations of the following structures (discuss any special circumstances that may change it)

(a) list (array based implementation)
(b) list (linked list implementation)
(c) binary tree
(d) binary search tree
(e) self-balancing binary search tree

16. Specify the worst case complexity of the add/insert and remove (item that “should be” removed) operations of the following structures

(a) queue (array based implementation)
(b) queue (singly linked list based implementation)
(c) queue (doubly linked list based implementation)
(d) priority queue (heap based implementation)

17. Write two methods that given a circularly linked list counts how many nodes are in that list.

18. Given a binary tree with 3 levels (level 0, level 1 and level 2) what is the largest number of nodes that the tree may contain? what is the smallest number of nodes that the tree may contain?

19. Given a heap with 3 levels (level 0, level 1 and level 2) what is the largest number of nodes that the heap may contain? what is the smallest number of nodes that the heap may contain?

20. Draw a binary tree for which the inorder and preorder traversals are as follows:

inorder: F E D B A C
preorder: B E F D C A

21. Given the following key values: 439, 340, 129, 342, 278, 947, 371 and a hash table stored in an array of size 10, show where in the array each of the keys is going to be stored if the has function is computed as h(x) = x % 10 and the following collision resolution method is used:

(a) separate chaining
(b) linear probing

22. Given a node in a binary tree, write a recursive method that computes the height of that node. The nodes do not store any height information. You may use pseudocode.