1. Fill in the runtimes for the following table.

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Insert</th>
<th>Search</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Queue</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Search Tree</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binary Heap</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hashtable</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10. Using the hash function $h(k) = k \mod 10$, draw the hash table using:

1. Chaining Approach

2. Open Addressing with linear Probing
3  Given the following tree write the preorder, postorder, and inorder traversal of nodes.

**Preorder:**

**Inorder:**

**Postorder:**

1. What is the runtime of the traversals?

2. Draw out the BST based on the following information:

Preorder: 15 8 11 9 20 17 16

Inorder: 8 9 11 15 16 17 20

4  Write code to Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists

Example:

Input: 1->2->4, 1->3->4

Output: 1->1->2->3->4->4

```python
def mergeTwoLists(l1, l2):
```

CS 1134 Final Exam  Review  Page 2
Given a non-empty binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

Symmetric tree examples:

```
      1     1     1
       / \   / \   / \
      2   2  2   2
     / \ / \ / \ / \
    3 4 4 3 3 3 3
```

Code:

```python
def is_symmetric(root):
```
6 Write the BFS and DFS going alphabetically when given multiple options.

**BFS:**

**DFS:**

7 Draw the corresponding max-heap from the given list: [50, 19, 36, 17, 3, 25, 1, 2, 7]

8 Using the above heap draw what happens step by step when you insert 23.
9 Using the above heap draw what happens step by step when you delete max.

10 Write a post-order traversal generator for a tree data structure class.

Code:

```python
def post_order(node):
```

Given a binary search tree and the lowest and highest boundaries as L and R, trim the tree so that all its values lies in the interval [L, R] (R >= L). You might need to change the root of the tree, so the result should return the new root of the trimmed binary search tree.

**Example 1:**

```
    15
   /   \
  8    20
  /      \
11    17
 /     / \
9     16
```

1. L = 11, R = 17

The tree will becomes:
```
    15
   /   \
  11    17
   /      \
  9    16
```

2. L = 16, R = 30

The tree will becomes:
```
    20
   /  \
  17
 / \
16
```
Code:
def trimBST(root,L,R):