1. Create a Huffman tree based on the character frequencies shown below.

<table>
<thead>
<tr>
<th>Character</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>126</td>
</tr>
<tr>
<td>Newline</td>
<td>53</td>
</tr>
<tr>
<td>A</td>
<td>102</td>
</tr>
<tr>
<td>B</td>
<td>60</td>
</tr>
<tr>
<td>C</td>
<td>40</td>
</tr>
<tr>
<td>D</td>
<td>15</td>
</tr>
<tr>
<td>E</td>
<td>21</td>
</tr>
<tr>
<td>F</td>
<td>80</td>
</tr>
<tr>
<td>G</td>
<td>50</td>
</tr>
</tbody>
</table>

2. Implement the private method leaves in the BinaryTree class. Your implementation must be recursive. See the example on the accompany page.
public class BinaryTree {
    //Implements a binary tree of strings
    private class Node {
        private Node left;
        private String data;
        private Node right;
        private Node(Node L, String d, Node R) {
            left = L;
            data = d;
            right = R;
        }
    }
    private Node root; //In a empty tree root is null

    //ASSUME VALUES HAVE BEEN PUT IN THE TREE

    public String leaves() {
        //if the tree is empty return the empty string otherwise
        //return a string of the data in the leaves in the tree
        return leaves(root);
    }

    private String leaves(Node r) {
        //if the r is null return the empty string
        //otherwise return a string of the data in the leaves of
        //the subtree rooted at r
        }
    }
For the tree shown below the function leaves should return the string
MNO IJ UVW PQ