BST Removal WS

Draw the tree shown to the right after each of the following: (Start over for each removal)
You can write SAME at any branch where the tree is unchanged

<table>
<thead>
<tr>
<th>Removing 14</th>
<th>Removing 10</th>
<th>Removing 8</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Tree Image" /></td>
<td><img src="image2" alt="Tree Image" /></td>
<td><img src="image3" alt="Tree Image" /></td>
</tr>
</tbody>
</table>

The project BSTCopyRemove has stubbed out functions:

```cpp
BSTNode<char>* copySubTree(BSTNode<char>* curNode)
char smallestValueFrom(BSTNode<char>* curNode)
BSTNode<char>* removeSmallestNode(BSTNode<char>* curNode)
```

Note that none of them are member functions – they are helper functions designed to be used by the BST member functions. Below is implementation strategy for each, pick either iterative or recursive and go with it.

It also has test code to generate the tree seen to the right and functions to test copying, finding the smallest value and removing the smallest value. Enable each test function as you are ready.

```cpp
BSTNode<char>* copySubTree(BSTNode<char>* curNode)
If the current node is null:
    return null
Node* newNode = new Node(curNode->value)
newNode's left = copySubTree(curNode's left)
newNode's right = copySubTree(curNode's right)
return the new node
```

Test with testCopy() function.