Question 1

Suppose we want to use a Binary Search Tree to store only keys (without any additional information), and we want to allow duplicate keys. Modify the TreeInsert algorithm to handle duplicate keys. Provide the updated algorithm and explain the changes.

Changes made: Add your explanation here.

Listing 1: Modified TreeInsert Algorithm

```python
def TreeInsert(root, key):
    if root is None:
        return Node(key)
    if key <= root.key:
        root.left = TreeInsert(root.left, key)
    else:
        root.right = TreeInsert(root.right, key)
    return root
```

Question 2

Describe the strategy to ensure duplicate keys are not always inserted on the same side. Use a boolean flag `goLeft` in each node and explain the changes.

Changes made: Add your explanation here.

Listing 2: Modified TreeInsert with goLeft Flag

```python
def TreeInsert(root, key):
    if root is None:
        return Node(key, goLeft=True)
    if key == root.key:
        if root.goLeft:
            root.left = TreeInsert(root.left, key)
        else:
            root.right = TreeInsert(root.right, key)
        root.goLeft = not root.goLeft
    elif key < root.key:
        root.left = TreeInsert(root.left, key)
    else:
        root.right = TreeInsert(root.right, key)
    return root
```

Question 3

Describe the strategy to randomly choose the subtree for duplicate keys. Explain the changes and provide the updated algorithm.

Changes made: Add your explanation here.

Listing 3: TreeInsert with Randomized Insertion

```
import random

def TreeInsert(root, key):
    if root is None:
        return Node(key)
    if key == root.key:
        if random.choice([True, False]):
            root.left = TreeInsert(root.left, key)
        else:
            root.right = TreeInsert(root.right, key)
    elif key < root.key:
        root.left = TreeInsert(root.left, key)
    else:
        root.right = TreeInsert(root.right, key)
    return root
```

Question 4

Propose a better strategy for handling duplicate keys and provide a complete algorithm with analysis.

Proposed Strategy: Describe your strategy here.

Listing 4: Proposed TreeInsert Algorithm

```python
def TreeInsert(root, key):
    # Your custom logic here
    pass
```