EXPLANATION FOR Q4

First, we create the structure of a tree node which contains the value of that node and the pointers to the two children nodes.

We implement circular deques using linked lists. This is similar to the implementation in Q2. The difference lies in the fact that the current implementation stores tree nodes.

Explanation of the queue functions:

(i) createqueue-

Worst Case Time Complexity = O(1).

It creates a node for the queue using dynamic memory allocation.

(ii) Push-

Worst Case Time Complexity = O(1).

It adds a node to the front of the queue

(iii) Pop-

Worst Case Time Complexity = O(1).

It removes a node from the back of the queue

(iv) Print-

Worst Case Time Complexity = O(n).

It prints all the nodes in the queue from the front to the back

(v) popRear-

Worst Case Time Complexity = O(1).

It removes a node from the back of the queue

Explanation of the tree functions:

(i) createNode-

Worst Case Time Complexity = O(1).

It creates a node for the tree using dynamic memory allocation.

(ii) insert-

Worst Case Time Complexity = O(n).

It inserts a node in the tree.

(iii) ListToBST-

Worst Case Time Complexity = O(n*n).

It traverses the whole array, makes tree nodes containing the values in the array, and inserts those tree nodes into a tree. This tree is returned in the end. The TC of each insert is O(n) and we are inserting n times. Hence, the overall TC is O(n*n).

(iv) f-

Worst Case Time Complexity = O(1).

It increases the value of the tree node by the sum amount and then assigns the value of the tree node to the sum, hence it updates the sum.

(v) change-

Worst Case Time Complexity = O(n).

It goes to each and every node in the tree and updates the value of each node. Since it visits every node, its time complexity is O(n).

(vi) ModifyBST-

Worst Case Time Complexity = O(n).

The difference between this function and the change function is that in change function, we need to send another parameter in every function call- the pointer to the sum. We wanted to construct the ModifyBST function such that we do not have to pass this parameter. This helps us implement the ModifyBST function exactly as asked to.

(v) levelorder-

Worst Case Time Complexity = O(n).

It traverse the whole tree and prints the nodes in a levelwise manner.