Lab Assignment -3

Name: Shubhang Tripathi
Enrollment No: 18114074
Batch: O3

Problem Statement -1:

Given the set of integers, write a C++ program to create a binary search tree (BST) and print all possible paths for it. You are not allowed to use subarray to print the paths. Convert the obtained BST into the corresponding AVL tree for the same input. AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property. Convert the obtained BST into the corresponding red-black tree for the same input. Red-Black Tree is a self-balancing Binary Search Tree (BST) where every node follows following rules.

1) Every node has a color either red or black.
2) Root of tree is always black.
3) There are no two adjacent red nodes (A red node cannot have a red parent or red child).
4) Every path from a node (including root) to any of its descendant NULL node has the same number of black nodes.

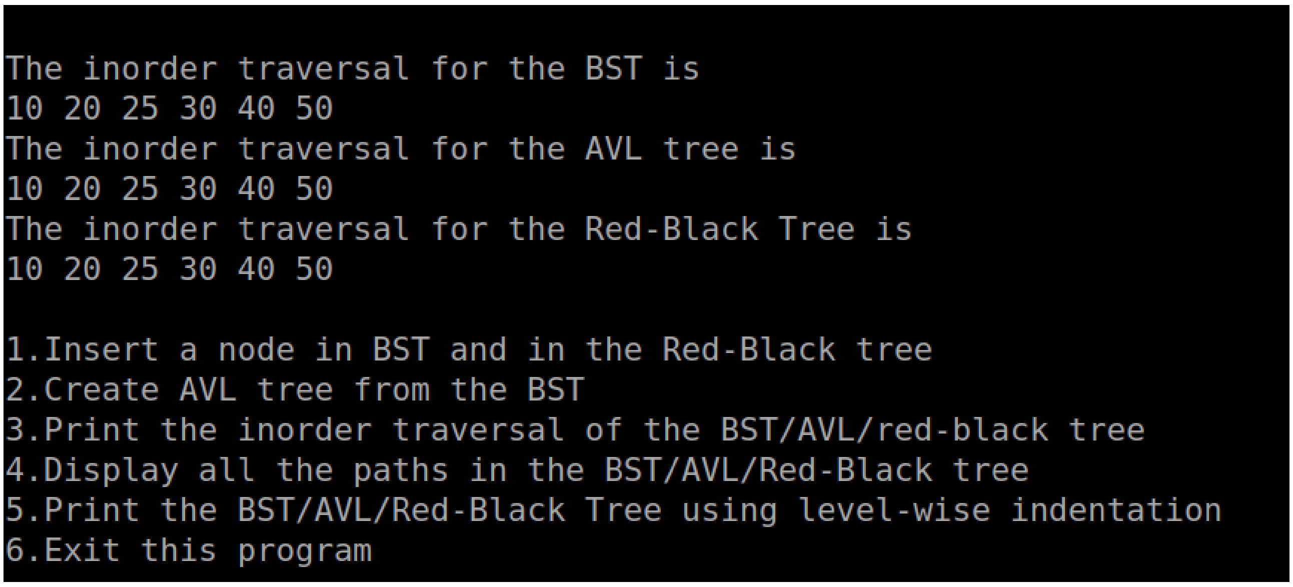
Write a menu driven program as follows:
1. To insert a node in the BST and in the red-black tree
2. To create AVL tree from the inorder traversal of the BST
3. To print the inorder traversal of the BST/AVL/red-black tree
4. To display all the paths in the BST/AVL tree/red-black tree
5. To print the BST/AVL tree/red-black Tree in the terminal using level-wise indentation (print color for red-black tree)
6. Exit

DATA STRUCTURES USED:

- Binary Search Trees
- AVL Trees
- Red-Black Trees
- Vectors

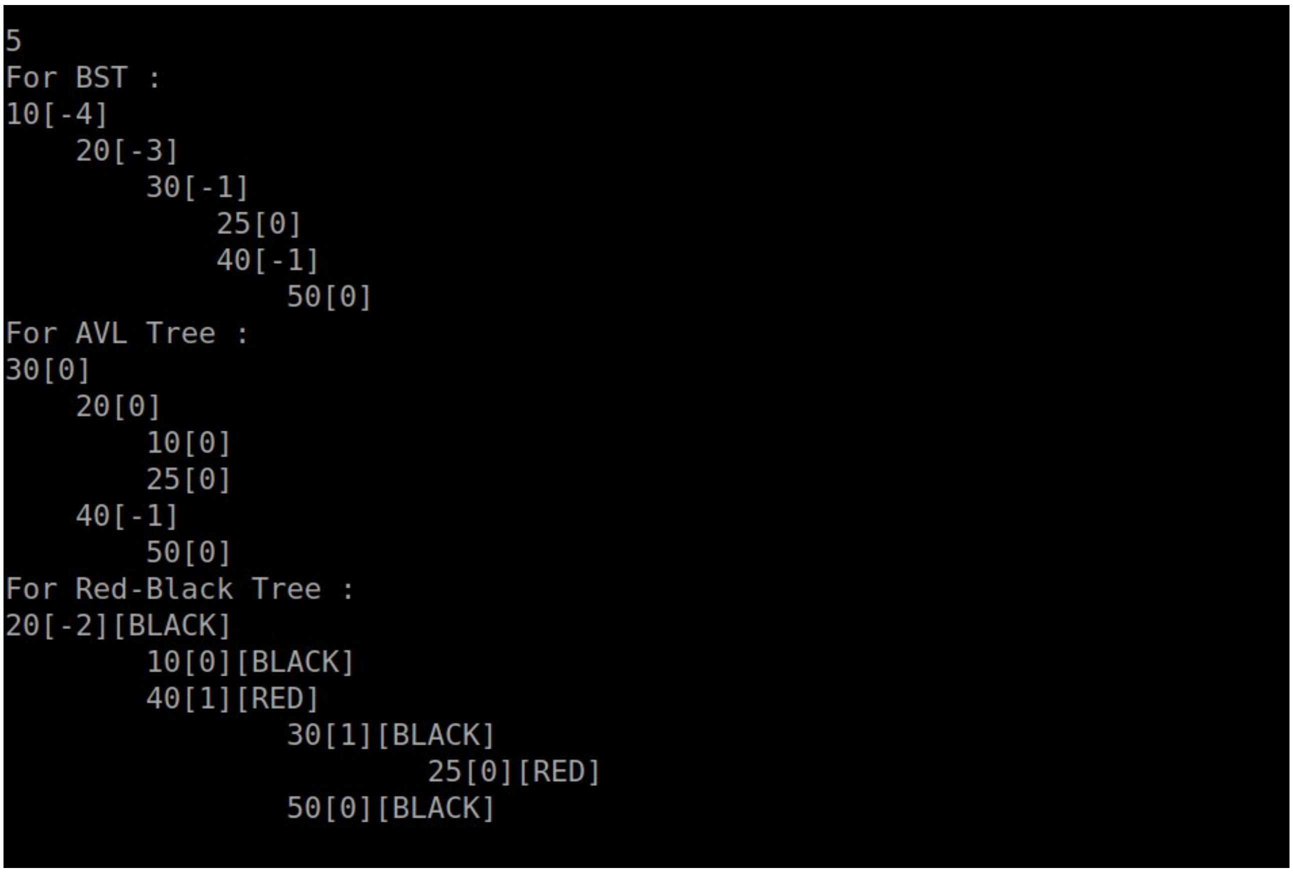
ALGORITHMS USED:

- Recursive algorithms for multiple functions like getHeight and printing paths
- Use of vectors to store the paths
- Using rotate functions to re-establish the property of Red-Black trees
- Using rotate functions to establish property of AVL trees

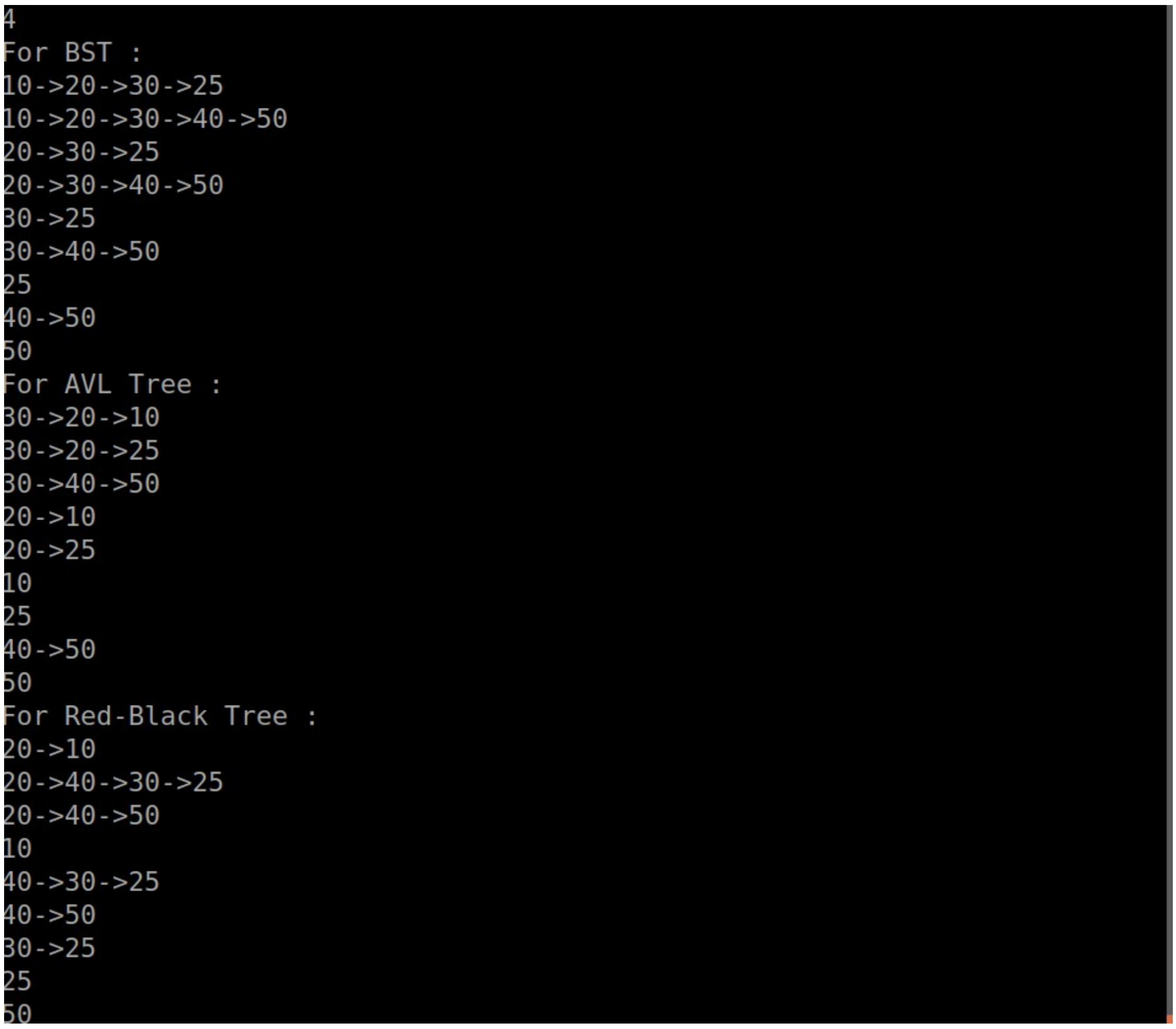


```
The inorder traversal for the BST is
10 20 25 30 40 50
The inorder traversal for the AVL tree is
10 20 25 30 40 50
The inorder traversal for the Red-Black Tree is
10 20 25 30 40 50

1.Insert a node in BST and in the Red-Black tree
2.Create AVL tree from the BST
3.Print the inorder traversal of the BST/AVL/red-black tree
4.Display all the paths in the BST/AVL/Red-Black tree
5.Print the BST/AVL/Red-Black Tree using level-wise indentation
6.Exit this program
```



```
5
For BST :
10[-4]
    20[-3]
        30[-1]
            25[0]
            40[-1]
                50[0]
For AVL Tree :
30[0]
    20[0]
        10[0]
        25[0]
    40[-1]
        50[0]
For Red-Black Tree :
20[-2][BLACK]
        10[0][BLACK]
        40[1][RED]
                30[1][BLACK]
                        25[0][RED]
                50[0][BLACK]
```



```
4
For BST :
10->20->30->25
10->20->30->40->50
20->30->25
20->30->40->50
30->25
30->40->50
25
40->50
50
For AVL Tree :
30->20->10
30->20->25
30->40->50
20->10
20->25
10
25
40->50
50
For Red-Black Tree :
20->10
20->40->30->25
20->40->50
10
40->30->25
40->50
30->25
25
50
```

Problem Statement -2:

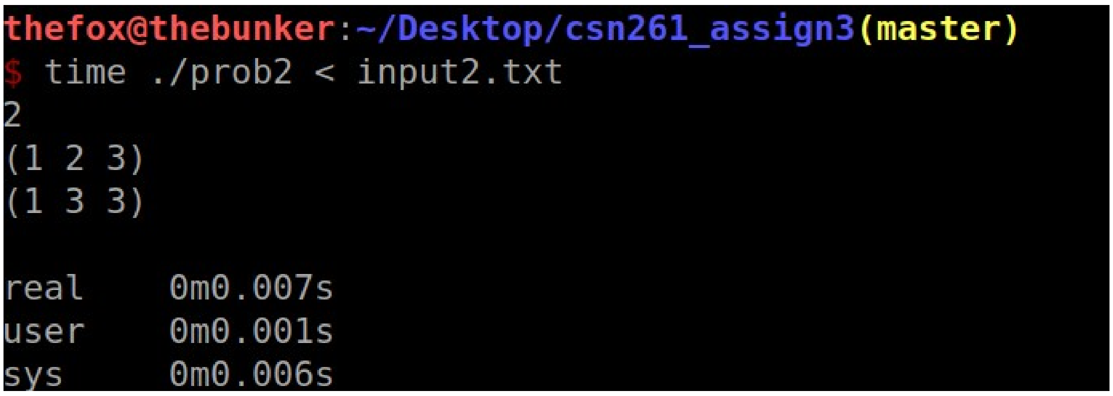
For a given sequence of positive integers $A_1, A_2, \ldots, A_N$ in decimal, find the triples (i, j, k), such that $1 \le i < j \le k \le N$ and $A_i \oplus A_{i+1} \oplus \ldots \oplus A_{j-1} = A_j \oplus A_{j+1} \oplus \ldots \oplus A_k$, where $\oplus$ denotes bitwise XOR.
This problem should be solved using dynamic programming approach and linked list data structures.
Print the number (count) of triples and list all the triplets in lexicographic order (each triplet in a new line).

DATA STRUCTURES USED:

- 2-Dimentional Array
- Linked List

ALGORITHMS USED:

- Dynamic Programming algorithm for getting xor
- Use of list to store the triplets in lexicographic order



```
thefox@thebunker:~/Desktop/csn261_assign3(master)
$ time ./prob2 < input2.txt
2
(1 2 3)
(1 3 3)

real	0m0.007s
user	0m0.001s
sys	0m0.006s
```