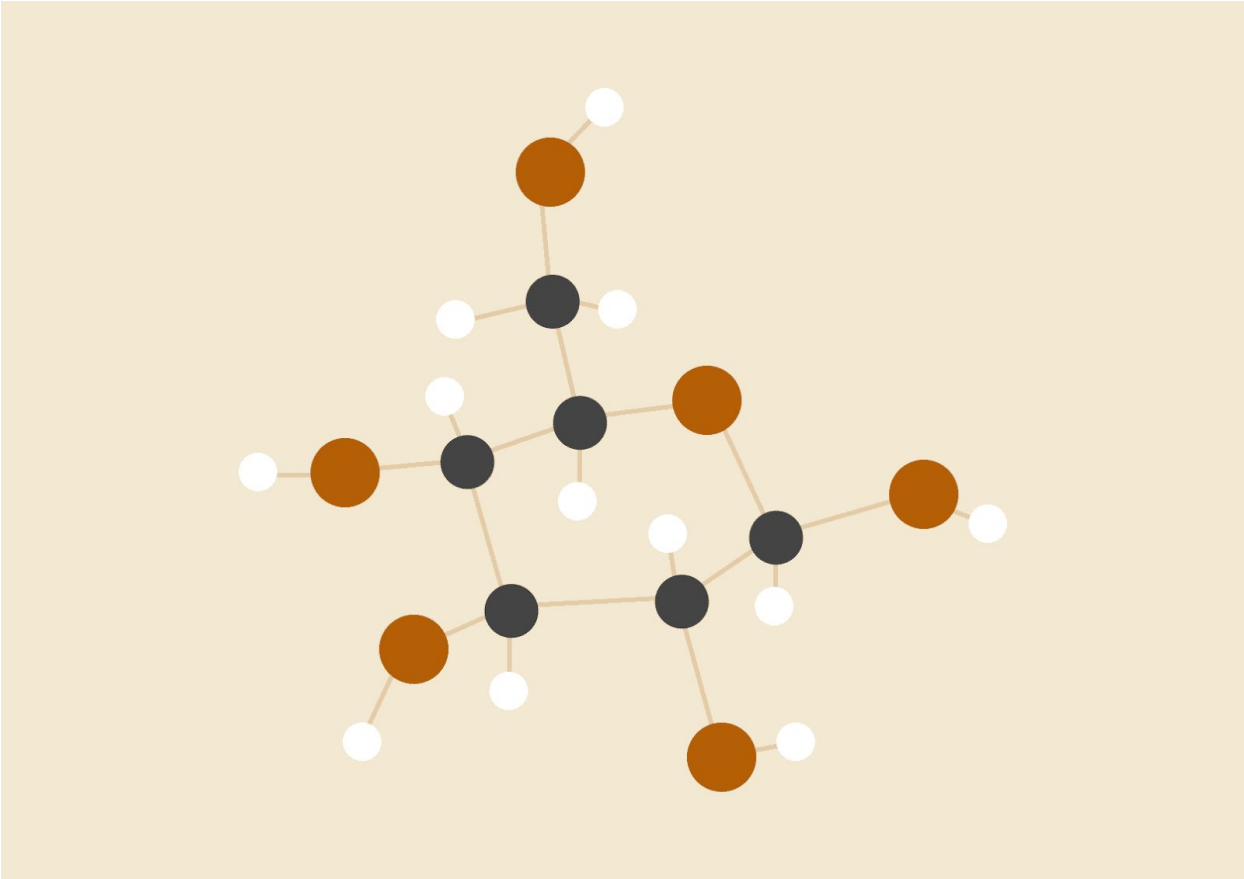
# ASSIGNMENT 3

Data Structures Laboratory




**Rishi Chordia**

18118052

BTech CSE

# 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

# Algorithms and Implementation:

In this question many different algorithms have been used.
Trees and Stacks data structures have been implemented.
We have two classes, one for BST and one for Red Black Tree and each class has its own set of operations.

- To create the BST simple insert operations have been used
- To create avl tree, we did an inorder traversal on BST and inserted the elements in the AVL Tree along with certain rotations
- To create the red black tree, appropriate insert function handling all cases of recolouring and rotation have been made
- To print all the paths of a tree, Depth First Search has been used which pushes the elements in a stack and pops it out whenever needed.
- Height of the trees are found using o(logn) algorithms
- A menu driven interface is provided with easy and user friendly usage
- Time has been calculate using the clock class in time.h
- GDB used for debugging

# Problem Statement 2:

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

**Input:**
**(a) Number of positive integers N.**
**(b) N space-separated integers $A_1$, $A_2$, ..., $A_N$.**

**Output:**
**Print the number (count) of triples and list all the triplets in lexicographic order (each triplet in a new line).**

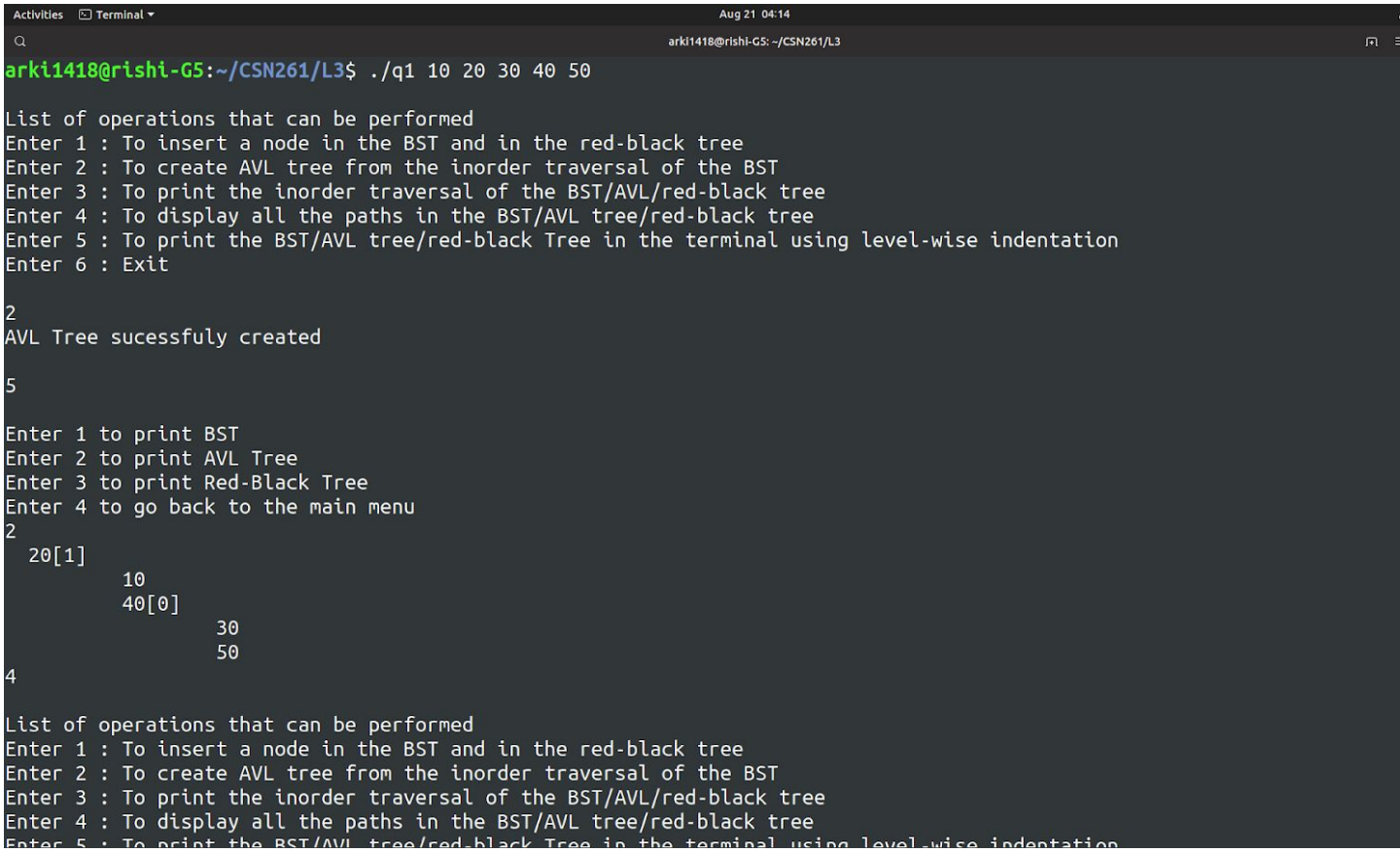
# Algorithms and Implementation:

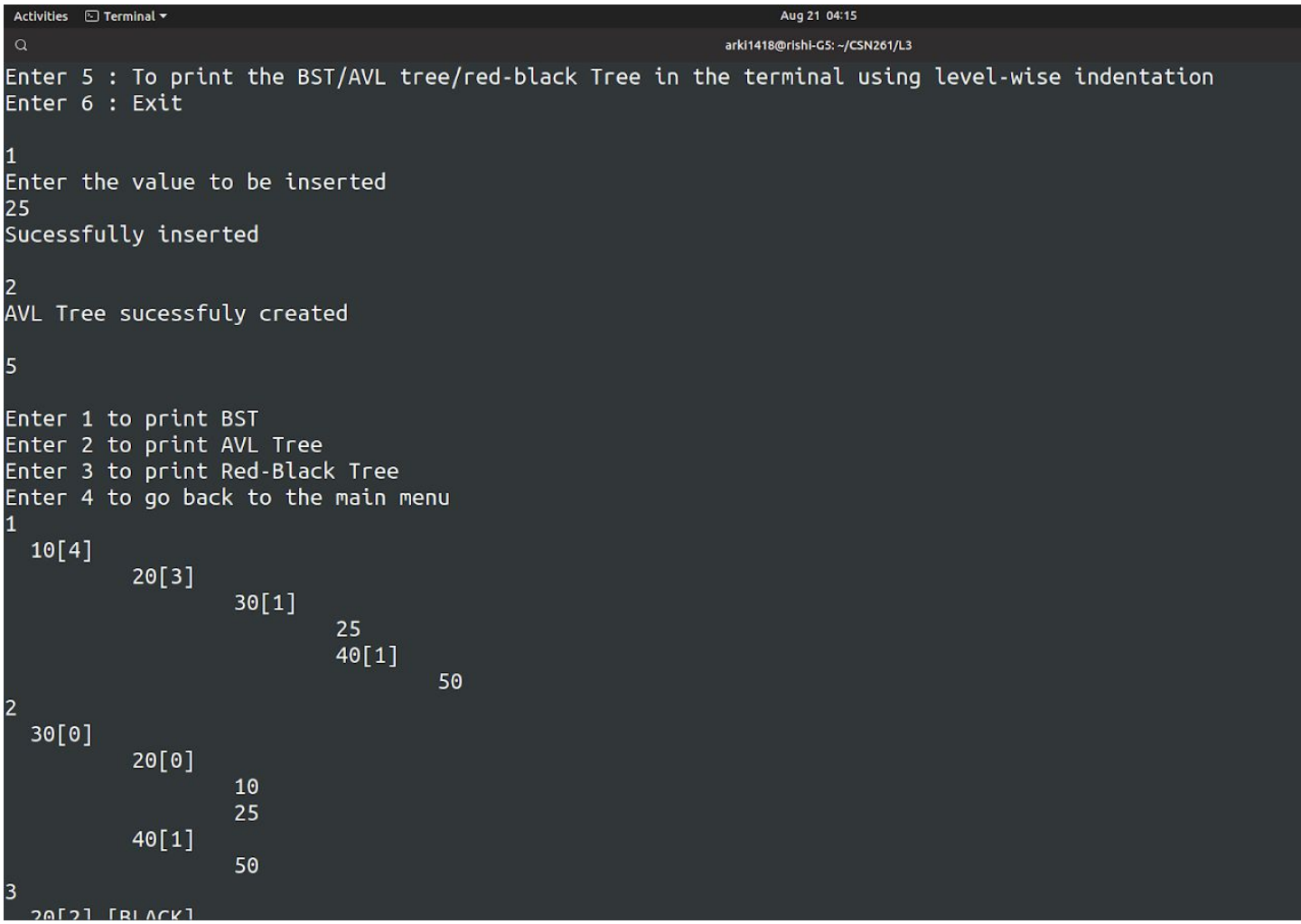
**In this question the basic approach was to**

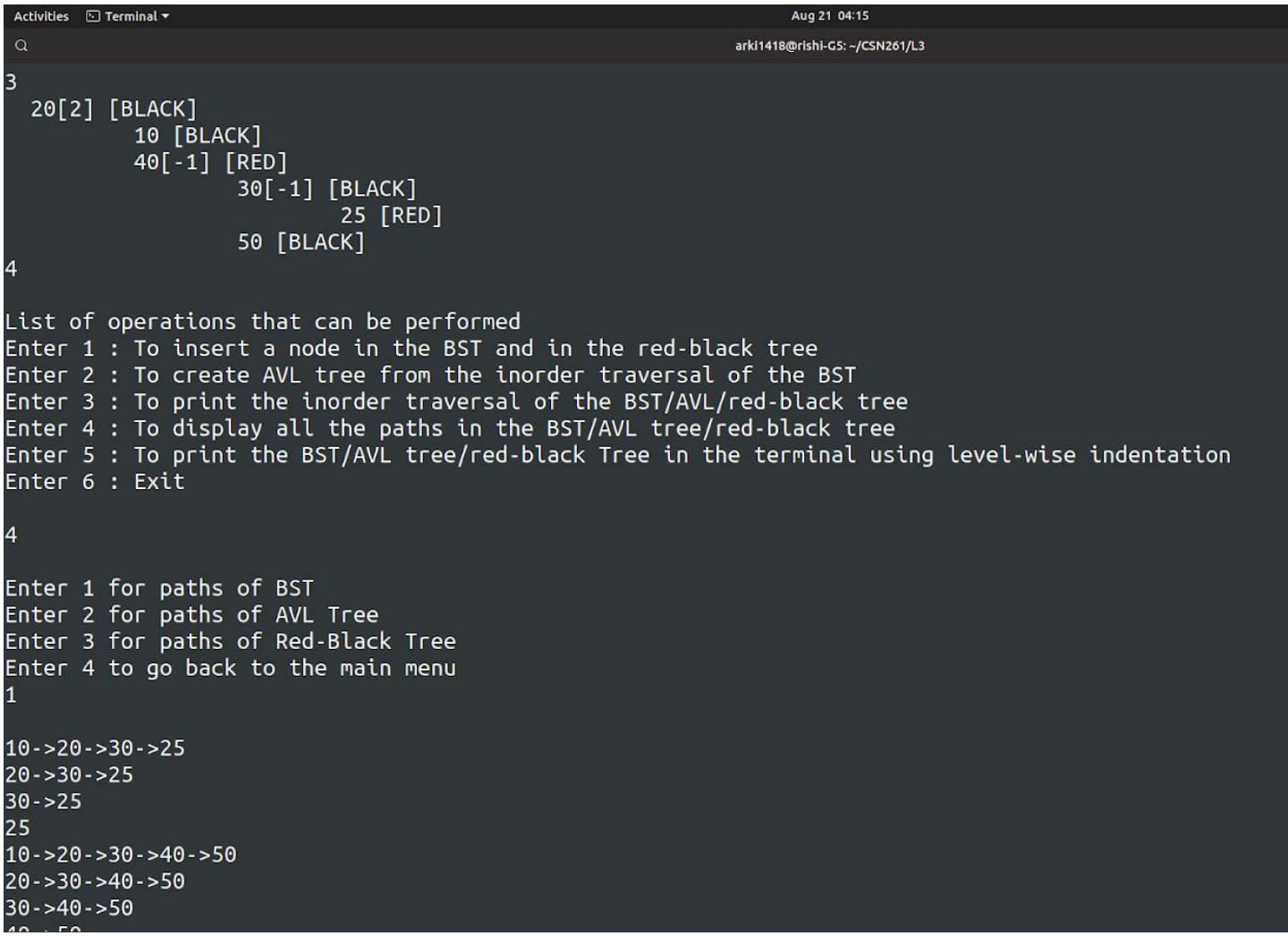
- **Linked List Data Structure has been used to store and operate on the given numbers.**
- **Dynamic Programming has been used to store the pre-xor's of the inserted numbers**
- **GDB was used for debugging**

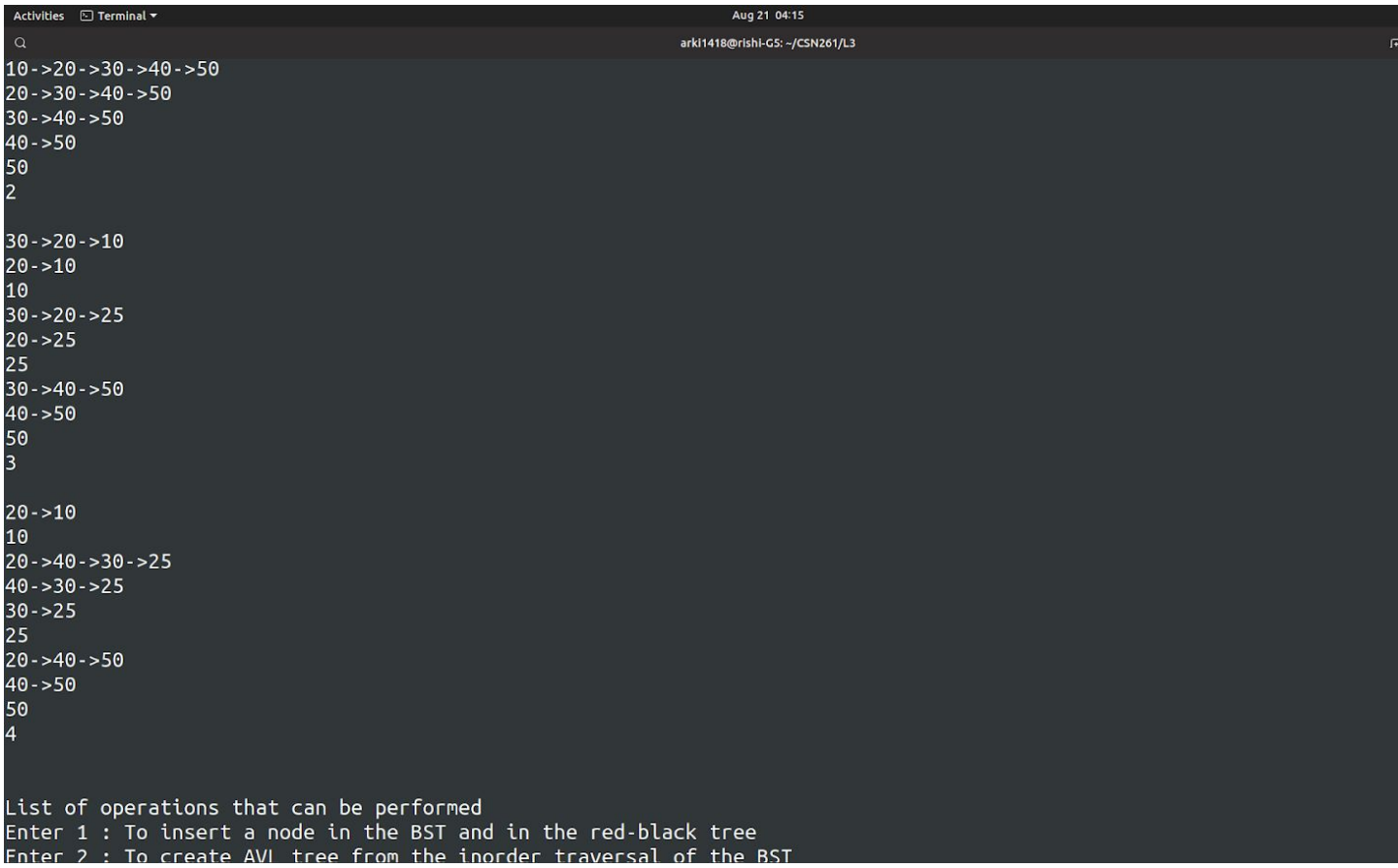
# Snapshots and Computation Time

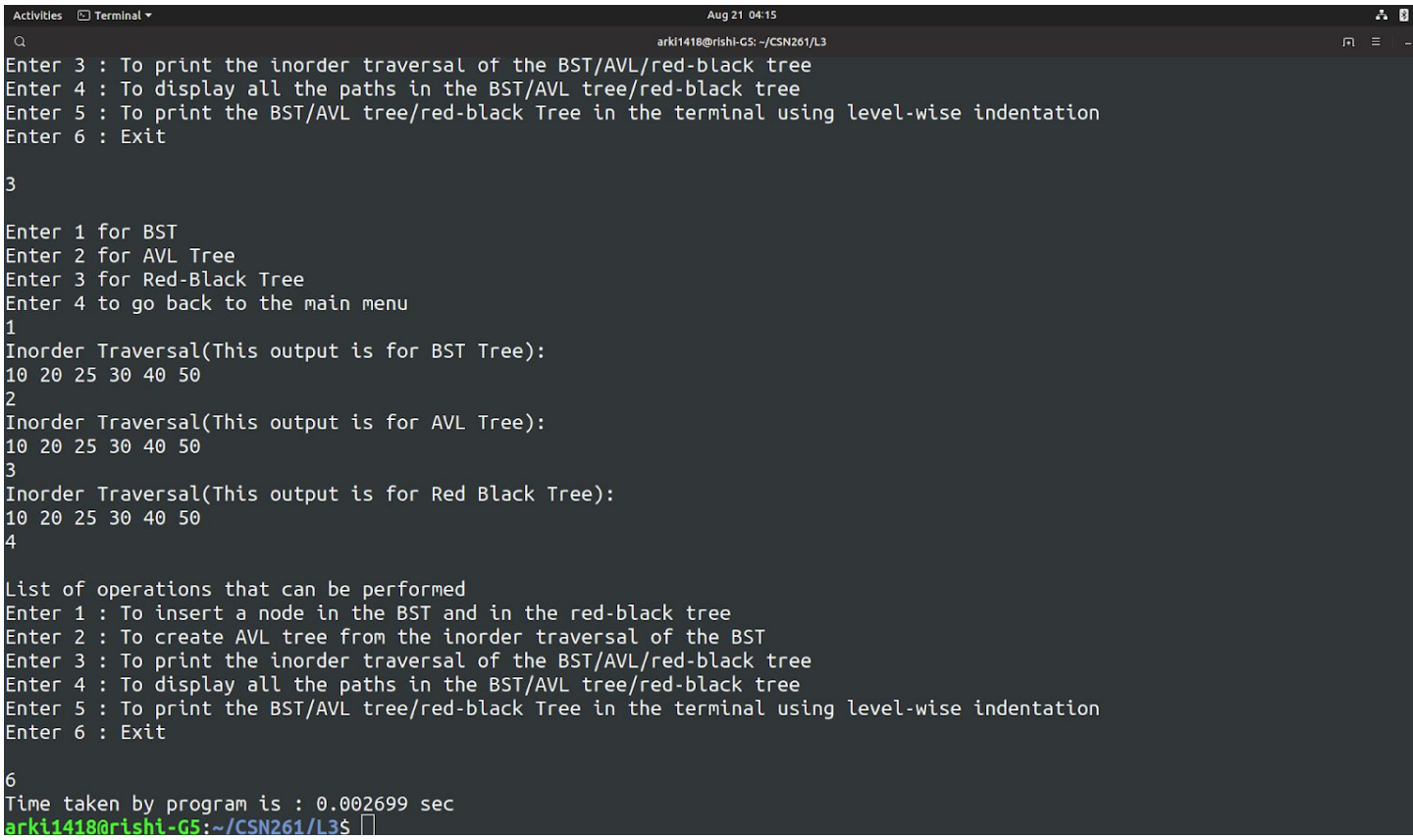
## Question 1:




Activities
Terminal
Aug 21 04:15
arki1418@rishi-G5: ~/CSN261/L3
Enter 5 : To print the BST/AVL tree/red-black Tree in the terminal using level-wise indentation
Enter 6 : Exit
1
Enter the value to be inserted
25
Sucessfully inserted
2
AVL Tree sucessfuly created
5
Enter 1 to print BST
Enter 2 to print AVL Tree
Enter 3 to print Red-Black Tree
Enter 4 to go back to the main menu
1
10[4]
20[3]
30[1]
25
40[1]
50
2
30[0]
20[0]
10
25
40[1]
50
3
20[2] [BLACK]

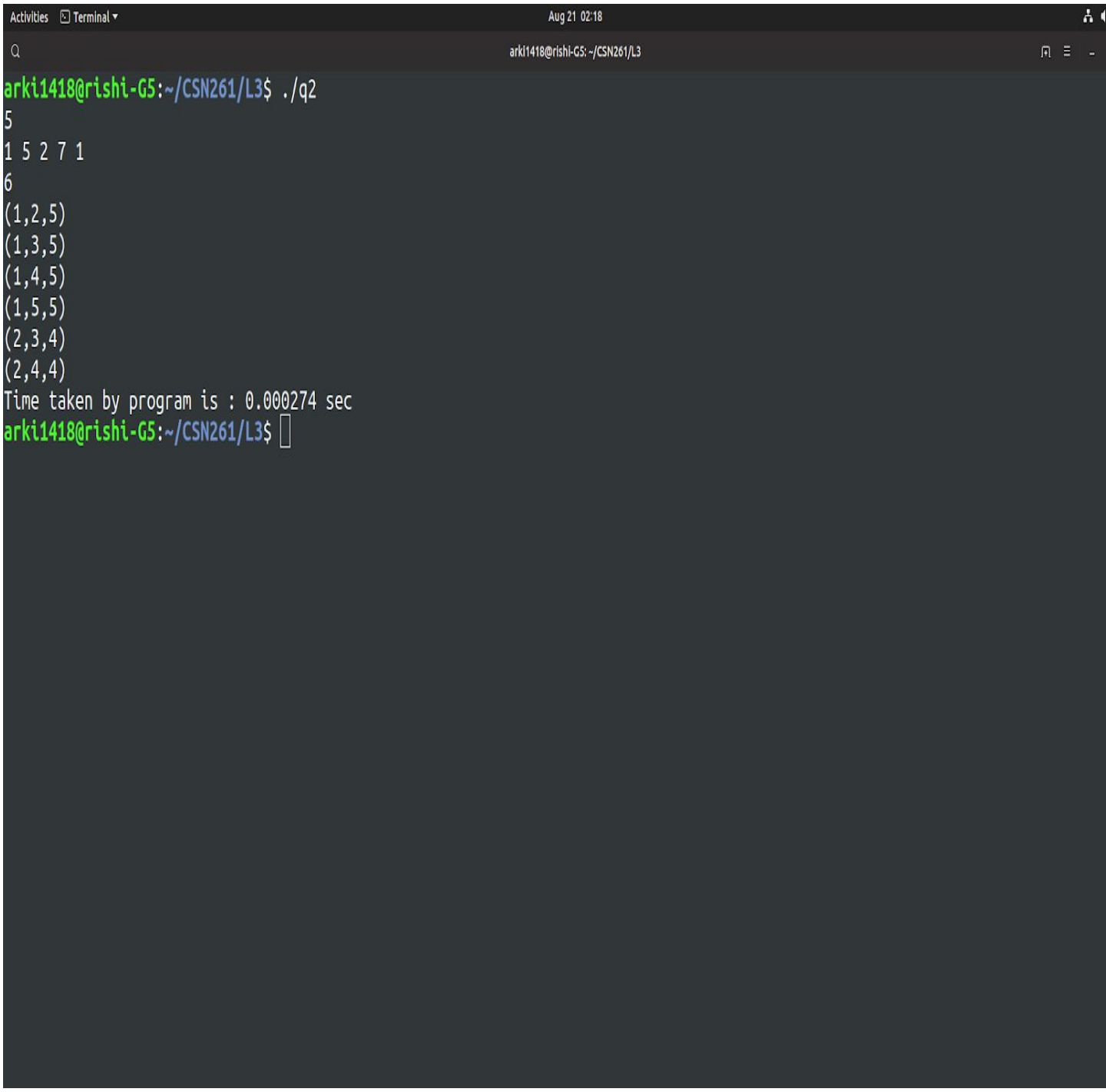

Activities
Terminal
Aug 21 04:15
arki1418@rishi-G5: ~/CSN261/L3
3
20[2] [BLACK]
10 [BLACK]
40[-1] [RED]
30[-1] [BLACK]
25 [RED]
50 [BLACK]
4
List of operations that can be performed
Enter 1 : To insert a node in the BST and in the red-black tree
Enter 2 : To create AVL tree from the inorder traversal of the BST
Enter 3 : To print the inorder traversal of the BST/AVL/red-black tree
Enter 4 : To display all the paths in the BST/AVL tree/red-black tree
Enter 5 : To print the BST/AVL tree/red-black Tree in the terminal using level-wise indentation
Enter 6 : Exit
4
Enter 1 for paths of BST
Enter 2 for paths of AVL Tree
Enter 3 for paths of Red-Black Tree
Enter 4 to go back to the main menu
1
10->20->30->25
20->30->25
30->25
25
10->20->30->40->50
20->30->40->50
30->40->50


Activities
Terminal
Aug 21 04:15
arkl1418@rishi-G5: ~/CSN261/L3
10->20->30->40->50
20->30->40->50
30->40->50
40->50
50
2
30->20->10
20->10
10
30->20->25
20->25
25
30->40->50
40->50
50
3
20->10
10
20->40->30->25
40->30->25
30->25
25
20->40->50
40->50
50
4
List of operations that can be performed
Enter 1 : To insert a node in the BST and in the red-black tree
Enter 2 : To create AVL tree from the inorder traversal of the BST




Final Computation Time of the above program is 0.002699 sec

# Question 2:


Activities
Terminal
Aug 21 02:18
arki1418@rishi-G5: ~/CSN261/L3
arki1418@rishi-G5:~/CSN261/L3$ ./q2
5
1 5 2 7 1
6
(1,2,5)
(1,3,5)
(1,4,5)
(1,5,5)
(2,3,4)
(2,4,4)
Time taken by program is : 0.000274 sec
arki1418@rishi-G5:~/CSN261/L3$