**Batch: C3 Roll No.: 16010123217**

**Experiment / assignment / tutorial No. 7**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:**  Implementation of BST & Binary tree traversal techniques. |

**Objective:** To Understand and Implement Binary Search Tree along with Insertion, Deletion and Preorder, Postorder and Inorder Traversal Techniques.

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| 1 | Explain the different data structures used in problem solving |

**Books/ Journals/ Websites referred:**

1. *Fundamentals Of Data Structures In C –* Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications –* Jean Paul Tremblay,

Paul G. Sorenson

1. *Data Structures A Pseudo Approach with C –* Richard F. Gilberg & Behrouz A. Forouzan
2. <https://www.geeksforgeeks.org/binary-tree-data-structure/>
3. <https://www.thecrazyprogrammer.com/2015/03/c-program-for-binary-search-tree-insertion.html>

**Abstract**:

**A tree** is a non- linear data structure used to represent hierarchical relationship existing among several data items. It is a finite set of one or more data items such that, there is a special data item called the root of the tree. Its remaining data items are partitioned into number of mutually exclusive subsets, each of which is itself a tree, and they are called subtrees.

**A binary tree** is a finite set of nodes. It is either empty or It consists a node called root with two disjoint binary trees-Left subtree, Right subtree. The Maximum degree of any node is 2

**A Binary Search Tree** is a node-based binary tree data structure in which the left subtree of a node contains only nodes with keys lesser than the node’s key. The right subtree of a node contains only nodes with keys greater than the node’s key. The left and right subtree each must also be a binary search tree.

**Related Theory: -**

**Algorithm: Preorder Traversal of BST**

Until all nodes are traversed −

Step 1 − Visit root node.

Step 2 − Recursively traverse left subtree.

Step 3 − Recursively traverse right subtree.

**Algorithm:Postorder Traversal of BST**

Until all nodes are traversed −

Step 1 − Recursively traverse left subtree.

Step 2 − Recursively traverse right subtree.

Step 3 − Visit root node.

**Algorithm: Inorder Traversal of BST**

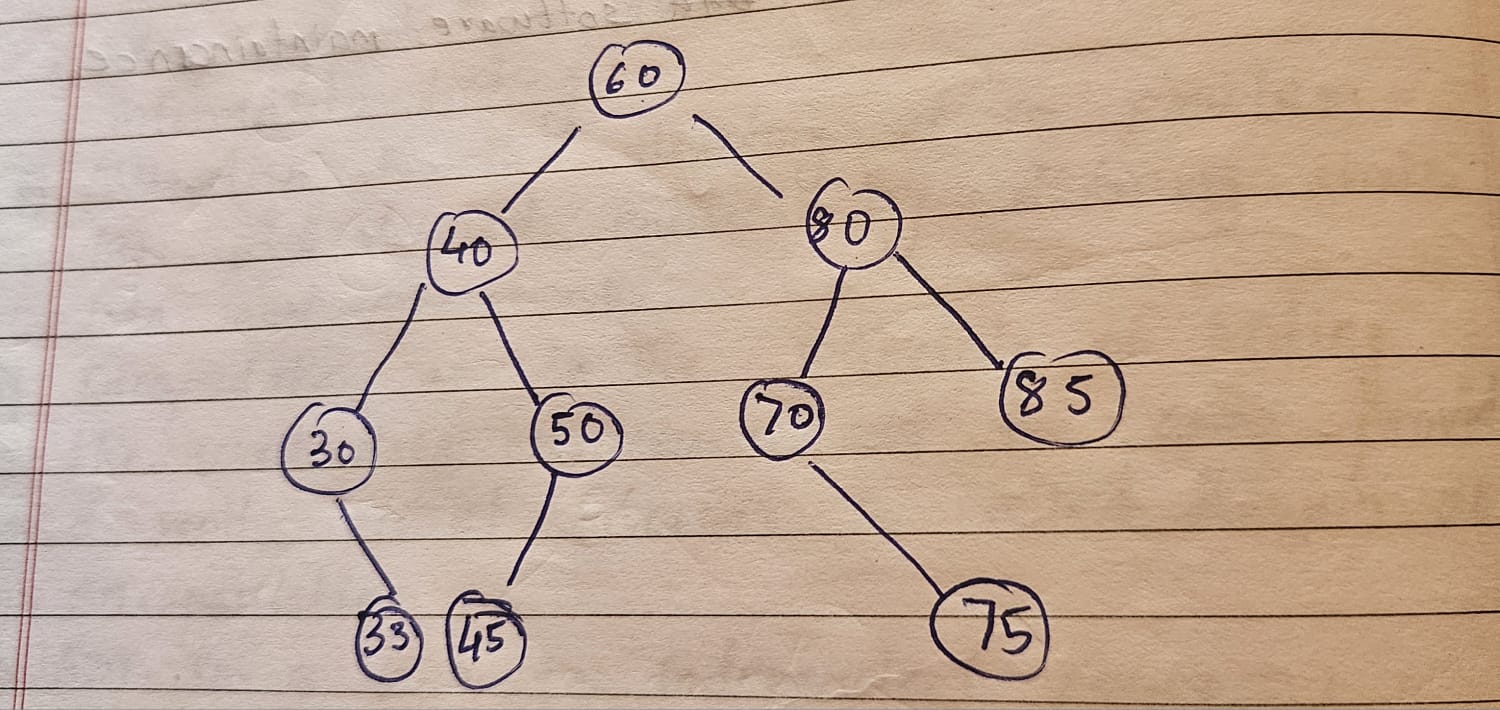
Until all nodes are traversed −

Step 1 − Recursively traverse left subtree.

Step 2 − Visit root node.

Step 3 − Recursively traverse right subtree

**An example BST :**



**Preorder Traversal:**

**60, 40, 30, 33, 50, 45, 80, 70, 75, 85**

**Postorder Traversal:**

**33, 30, 45, 50, 40, 75, 70, 85, 80, 60**

**Inorder Traversal:**

**30, 33, 40, 45, 50, 60, 70, 75, 80, 85**

**Algorithm for insertion into a BST:**

Algorithm InsertTree(TreeType tree, ElementType element)

if tree is EmptyTree

tree ← createNode(element)

else if element < tree->Element

InsertTree(tree->Left, element)

else if element > tree->Element

InsertTree(tree->Right, element)

else

print "Element already exists"

**Program source code for Implementation of BST insertion & Binary tree traversal techniques :**

*//16010123217 Om Thanage*

#include <stdio.h>

#include <stdlib.h>

struct node {

    int data;

    struct node \*left;

    struct node \*right;

};

struct node \*root = NULL;

void insert(int item) {

    struct node \*temp, \*ptemp, \*ctemp;

    temp = (struct node \*)malloc(sizeof(struct node));

    if (temp == NULL) {

        printf("Can't insert\n");

    } else {

        temp->data = item;

        temp->left = NULL;

        temp->right = NULL;

        if (root == NULL) {

            root = temp;

        } else {

            ptemp = NULL;

            ctemp = root;

            while (ctemp != NULL) {

                ptemp = ctemp;

                if (item < ctemp->data) {

                    ctemp = ctemp->left;

                } else {

                    ctemp = ctemp->right;

                }

            }

            if (item < ptemp->data) {

                ptemp->left = temp;

            } else {

                ptemp->right = temp;

            }

        }

        printf("Node Inserted\n");

    }

}

*// Preorder traversal (Root, Left, Right)*

void preorder(struct node \*node) {

    if (node != NULL) {

        printf("%d ", node->data);

        preorder(node->left);

        preorder(node->right);

    }

}

*// Inorder traversal (Left, Root, Right)*

void inorder(struct node \*node) {

    if (node != NULL) {

        inorder(node->left);

        printf("%d ", node->data);

        inorder(node->right);

    }

}

*// Postorder traversal (Left, Right, Root)*

void postorder(struct node \*node) {

    if (node != NULL) {

        postorder(node->left);

        postorder(node->right);

        printf("%d ", node->data);

    }

}

int main() {

    int item;

    printf("Enter the number of elements to be Inserted:\n");

    int n;

    scanf("%d", &n);

    for(int i  = 0; i<n; i++){

        printf("Enter the element %d:\t", i + 1 );

        scanf("%d", &item);

        insert (item);

    }

    printf("\nPreorder traversal: ");

    preorder(root);

    printf("\nInorder traversal: ");

    inorder(root);

    printf("\nPostorder traversal: ");

    postorder(root);

    int choice = 1;

    while(choice == 1){

        printf("\nDo you want to insert? (Press 0 to No, Press 1 to Yes)");

        scanf("%d", &choice);

        if(choice == 1){

            printf("Enter the element to be inserted:\n");

            scanf("%d", &item);

            insert (item);

            printf("\nPreorder traversal after inserting %d: ", item);

            preorder(root);

            printf("\nInorder traversal after inserting %d: ", item);

            inorder(root);

            printf("\nPostorder traversal after inserting %d: ", item);

            postorder(root);

        }

        else{

            printf("Thank you are using");

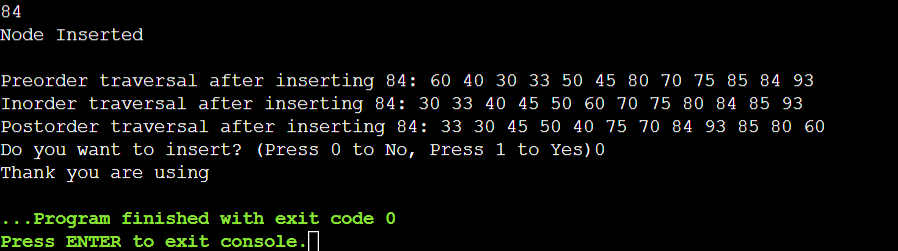
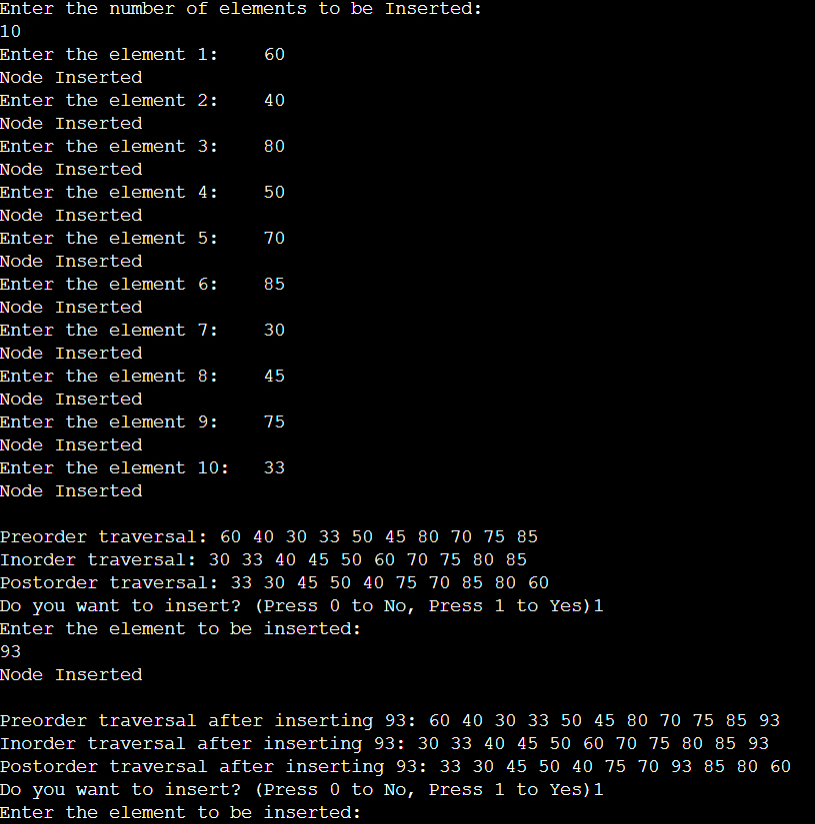
        }

    }

    return 0;

}

**Output Screenshots for Each Operation:**



**Conclusion:-**

**Post Lab Questions:**

1. **Write an ADT for tree data structure**

**Ans. Value Definition:**

Abstract typedef TreeType(ElementType ele)

Condition : none;

**Operator definition:**

1. Abstract TreeType CreateEmptyTree()

Precondition: None

Postcondition: An empty binary search tree (BST) is created.

2. Abstract TreeType InsertTree(TreeType Tree, ElementType Element)

Precondition: The tree is not null.

Postcondition: The element is inserted into the binary search tree at the appropriate position while maintaining the BST property (left < root < right).

3. Abstract Boolean SearchTree(TreeType Tree, ElementType Element)

Precondition: The tree is not null.

Postcondition: The function returns True if the element is found in the tree; otherwise, it returns False.

4. Abstract void PreOrderTraversal(TreeType Tree)

Precondition: The tree is not null.

Postcondition: The elements of the tree are printed or accessed in preorder (root, left, right).

5. Abstract void InOrderTraversal(TreeType Tree)

Precondition: The tree is not null.

Postcondition: The elements of the tree are printed or accessed in inorder (left, root, right), ensuring a sorted order.

6. Abstract void PostOrderTraversal(TreeType Tree)

Precondition: The tree is not null.

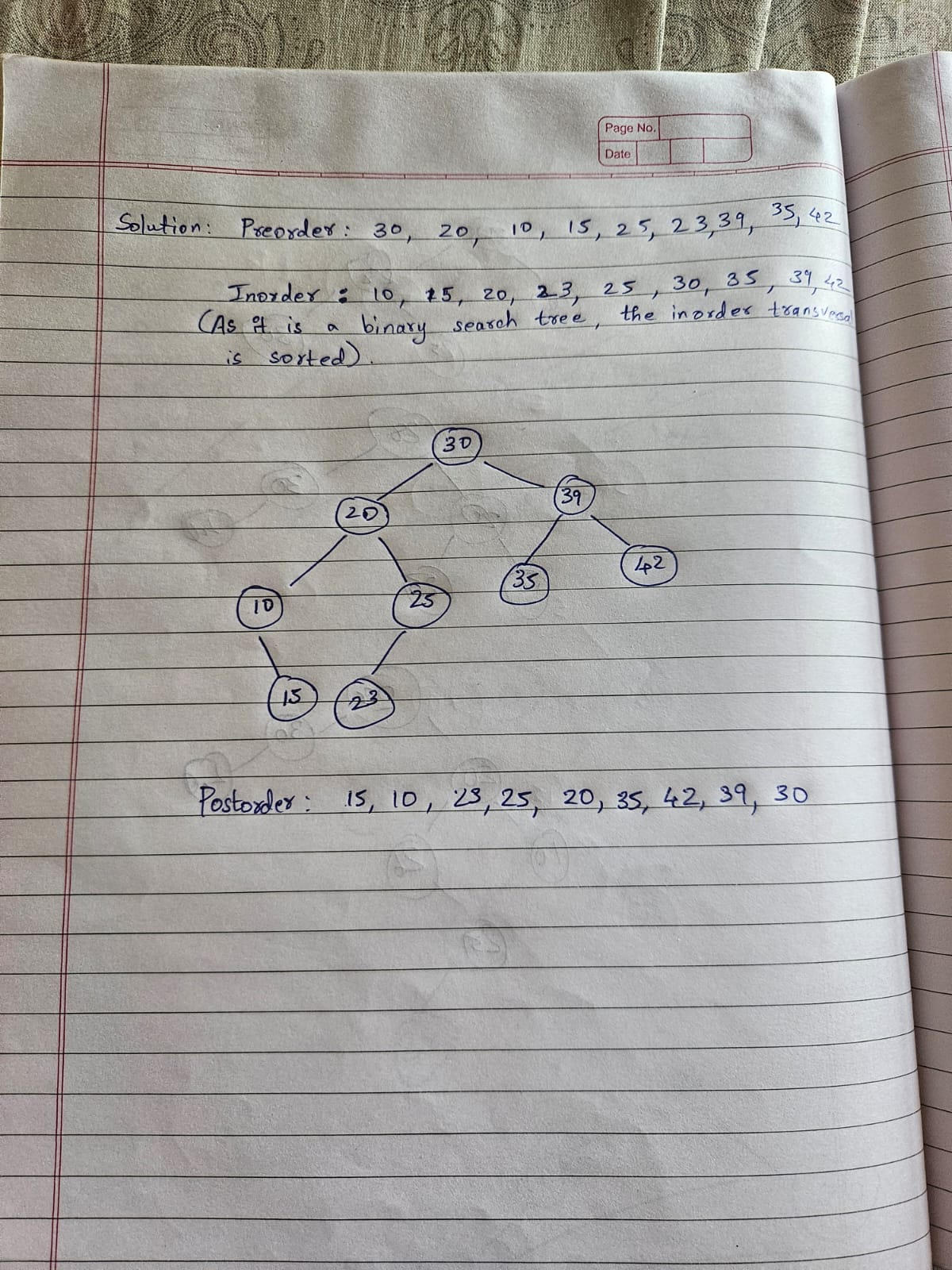
Postcondition: The elements of the tree are printed or accessed in postorder (left, right, root)

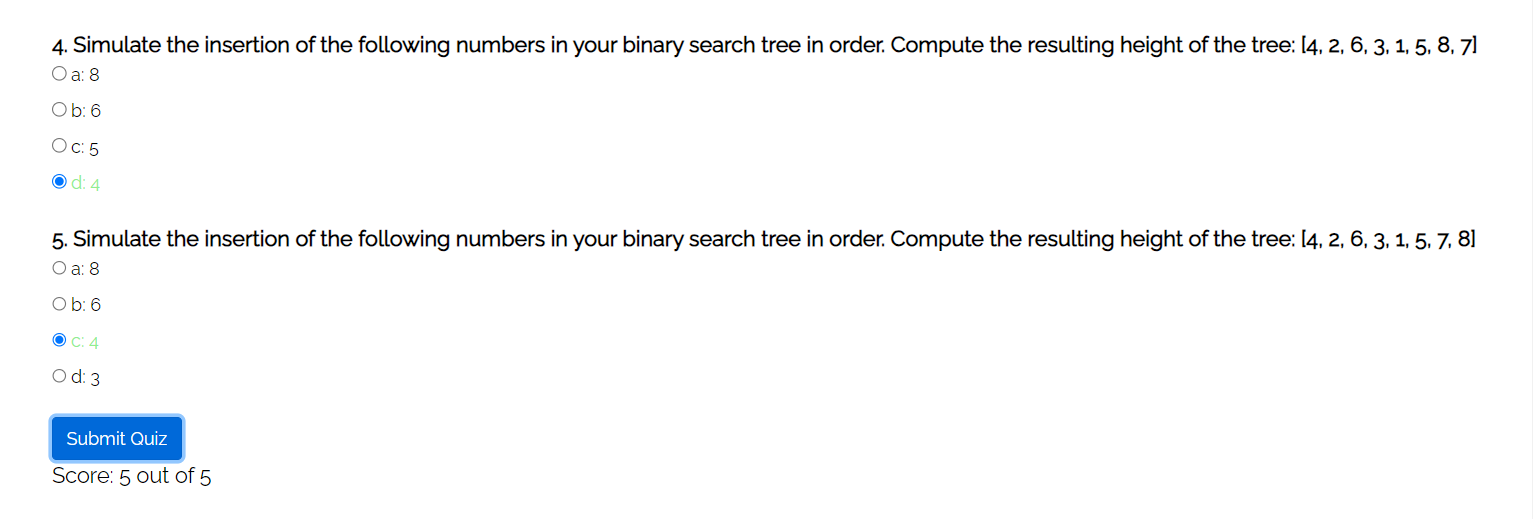
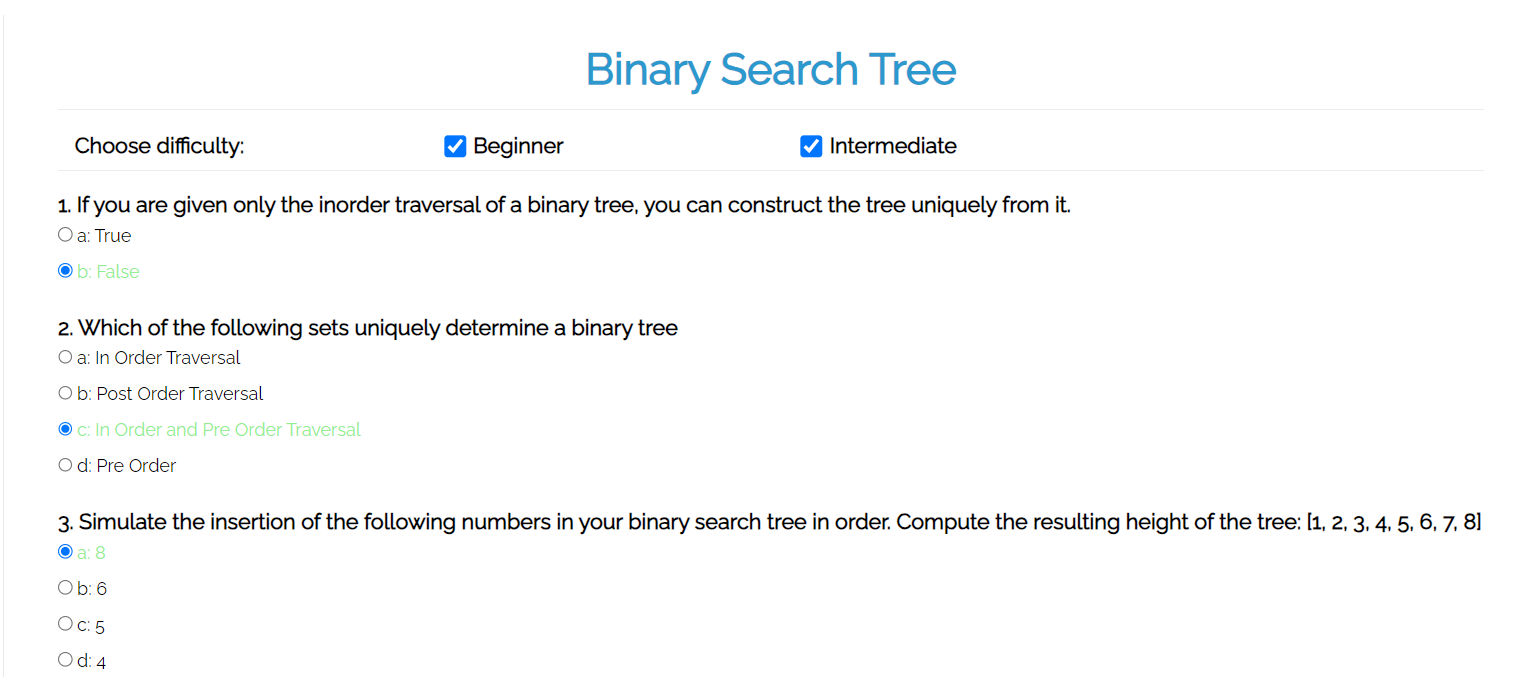
1. **What are the consequences of inserting elements in ascending or descending order into a BST?**

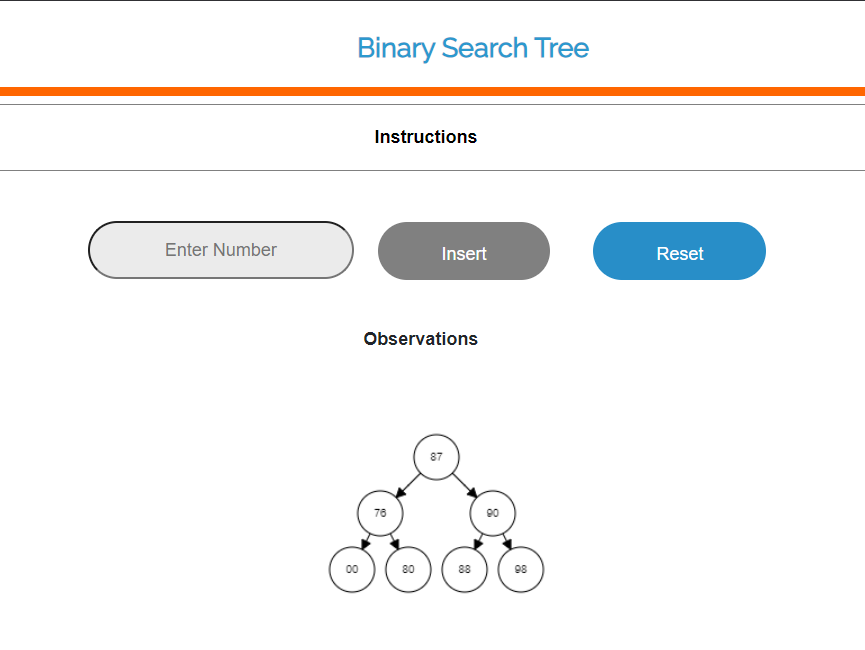
**Ans.** The consequences of inserting in ascending order would be that the all elements will be at the right side with no left child making it a linked list instead of balanced binary tree. All some operations like search would take more time than usual binary search tree.

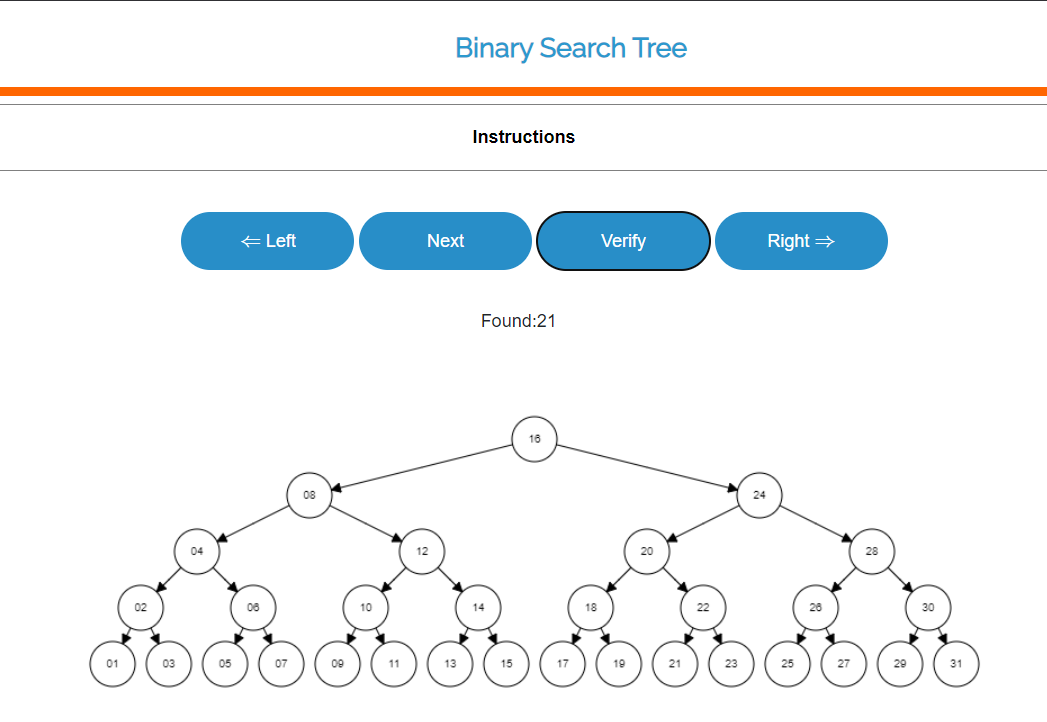
Similarly, in descending order all the elements will go to the left with no right chils making it a linked list instead of a binary search tree.

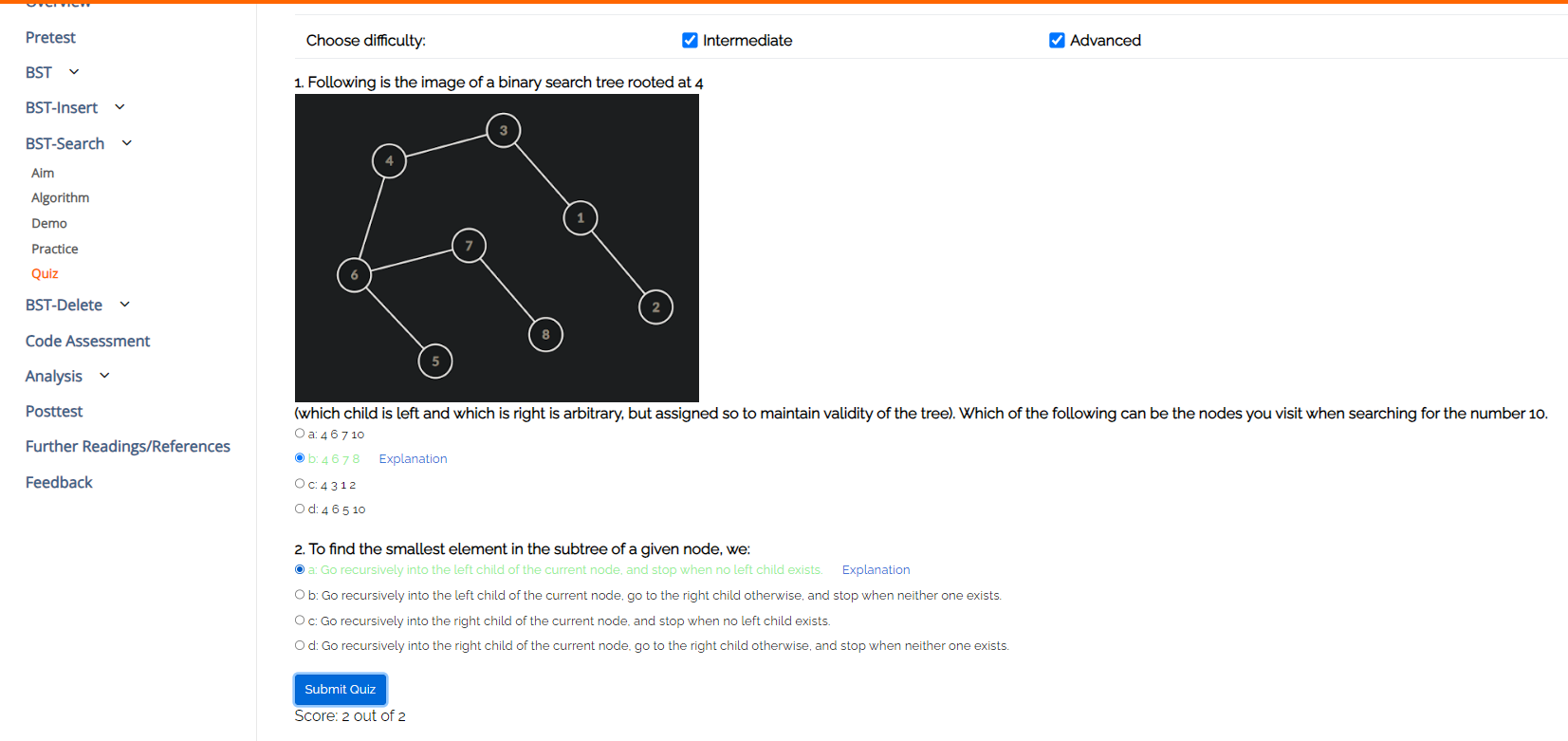
1. **The preorder traversal sequence of a binary search tree is 30, 20, 10, 15, 25, 23, 39, 35, 42. Construct a balanced Binary Search Tree and perform the Postorder Traversal for the same.**

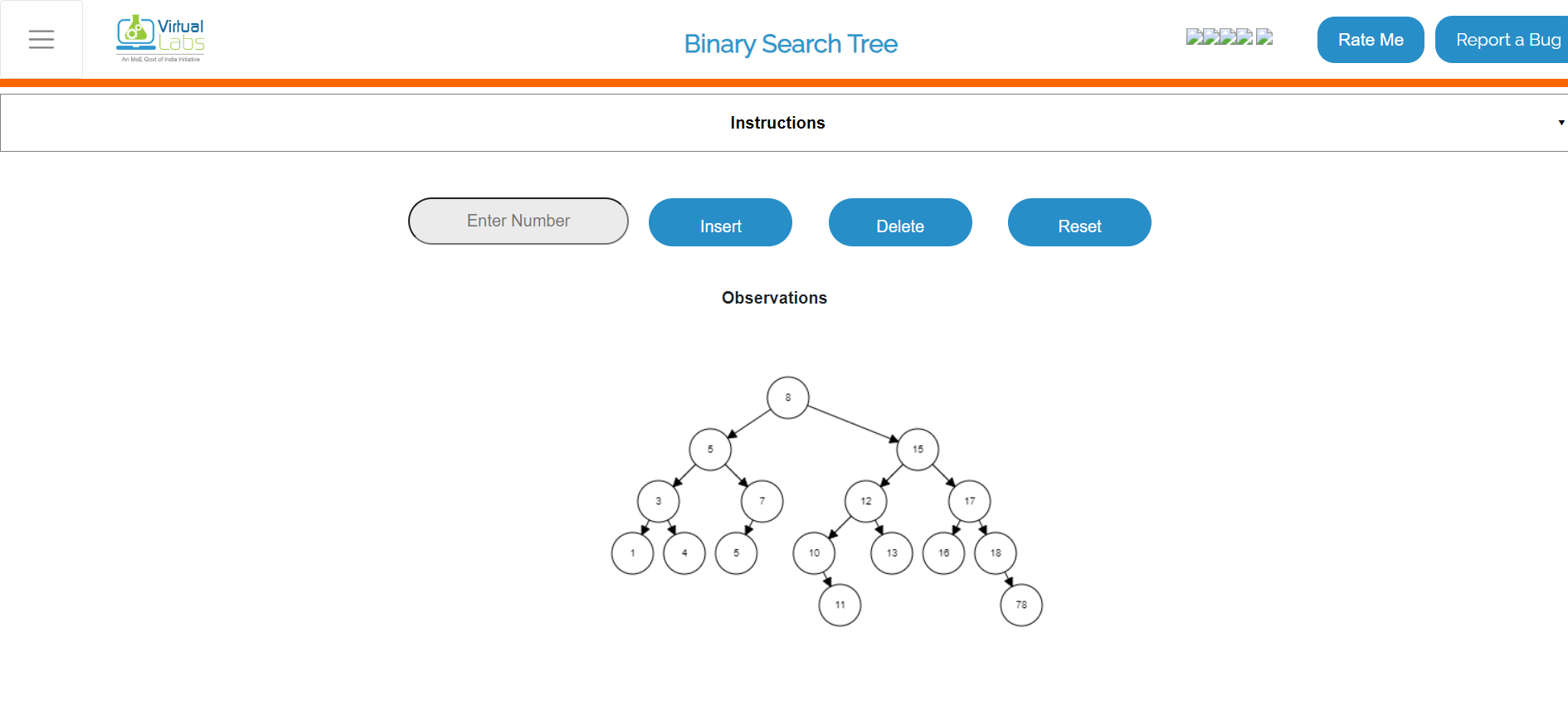


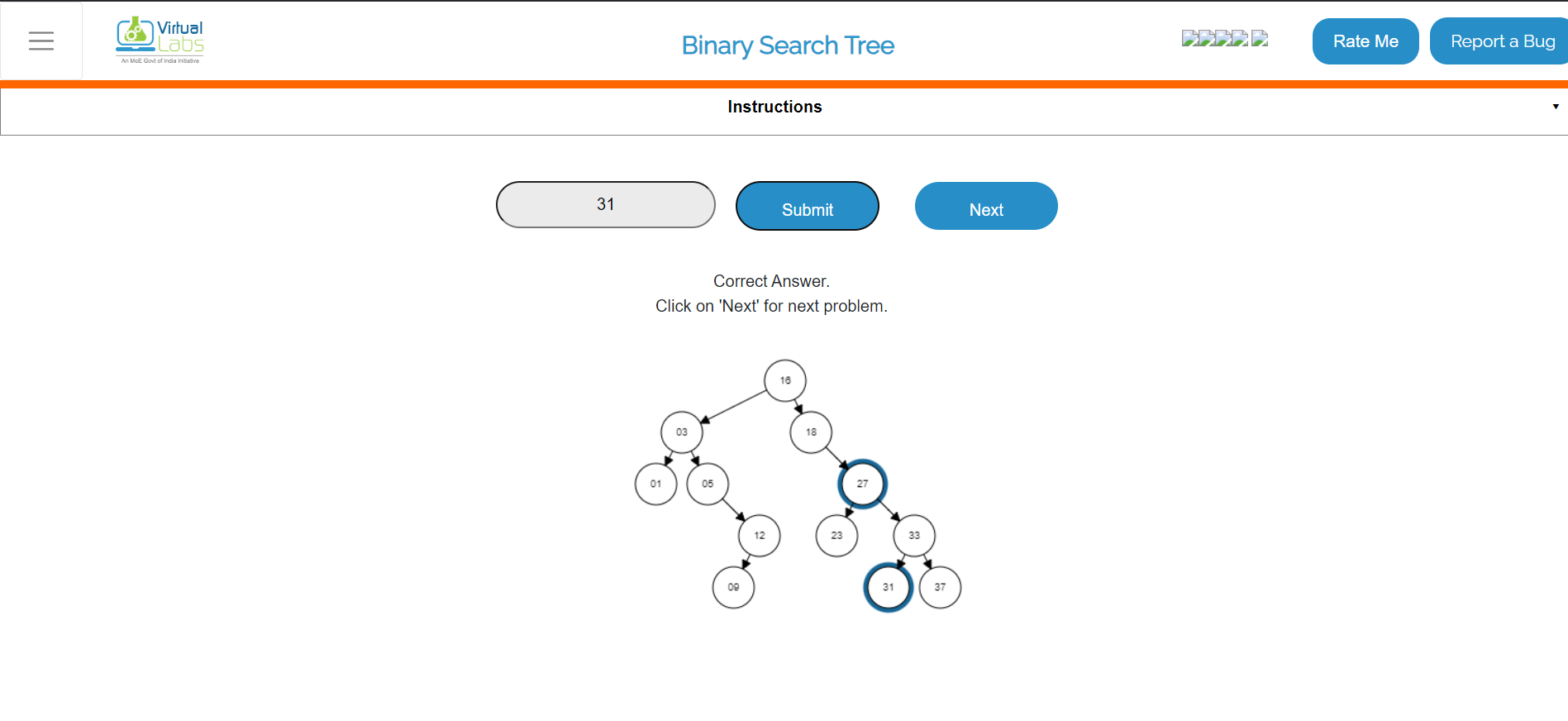
**Vlab:  
  
**

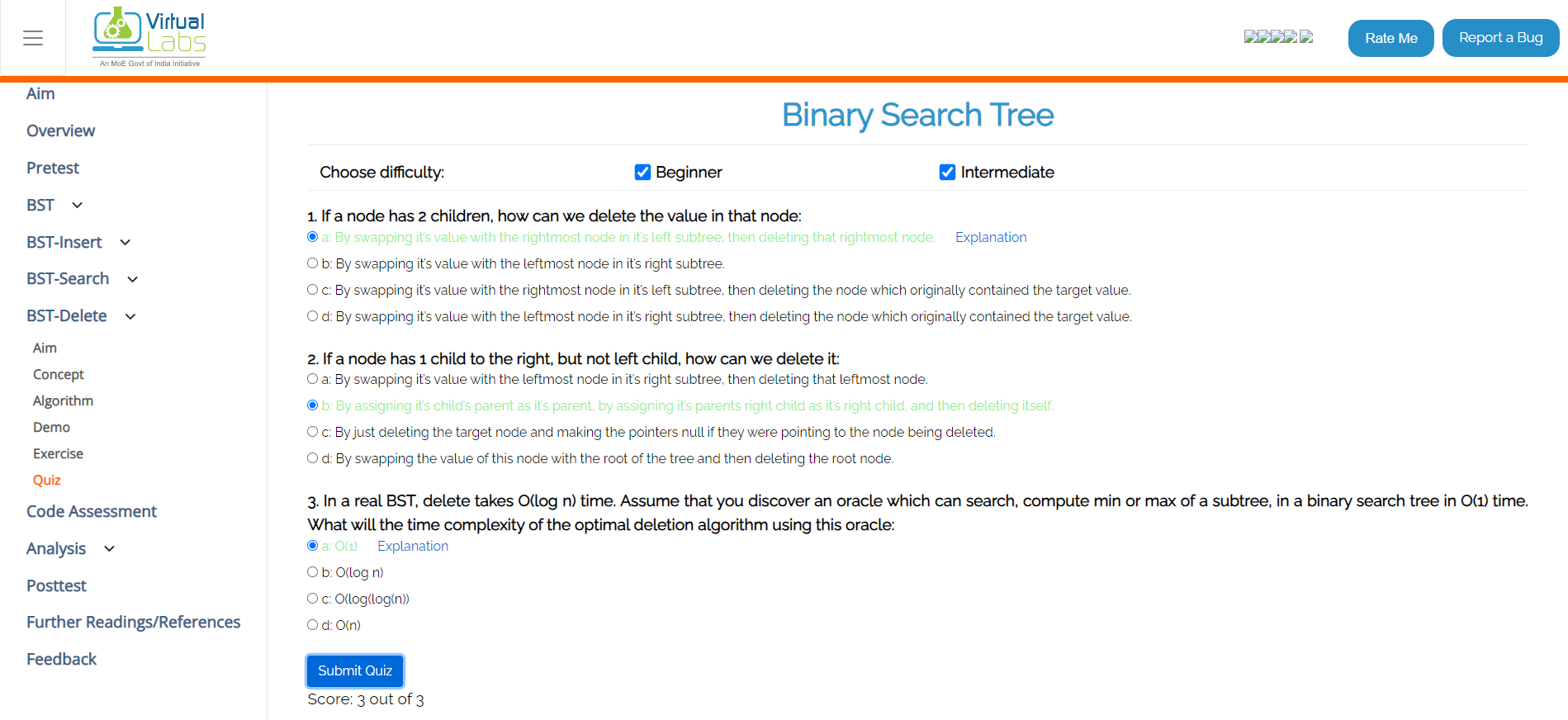
****

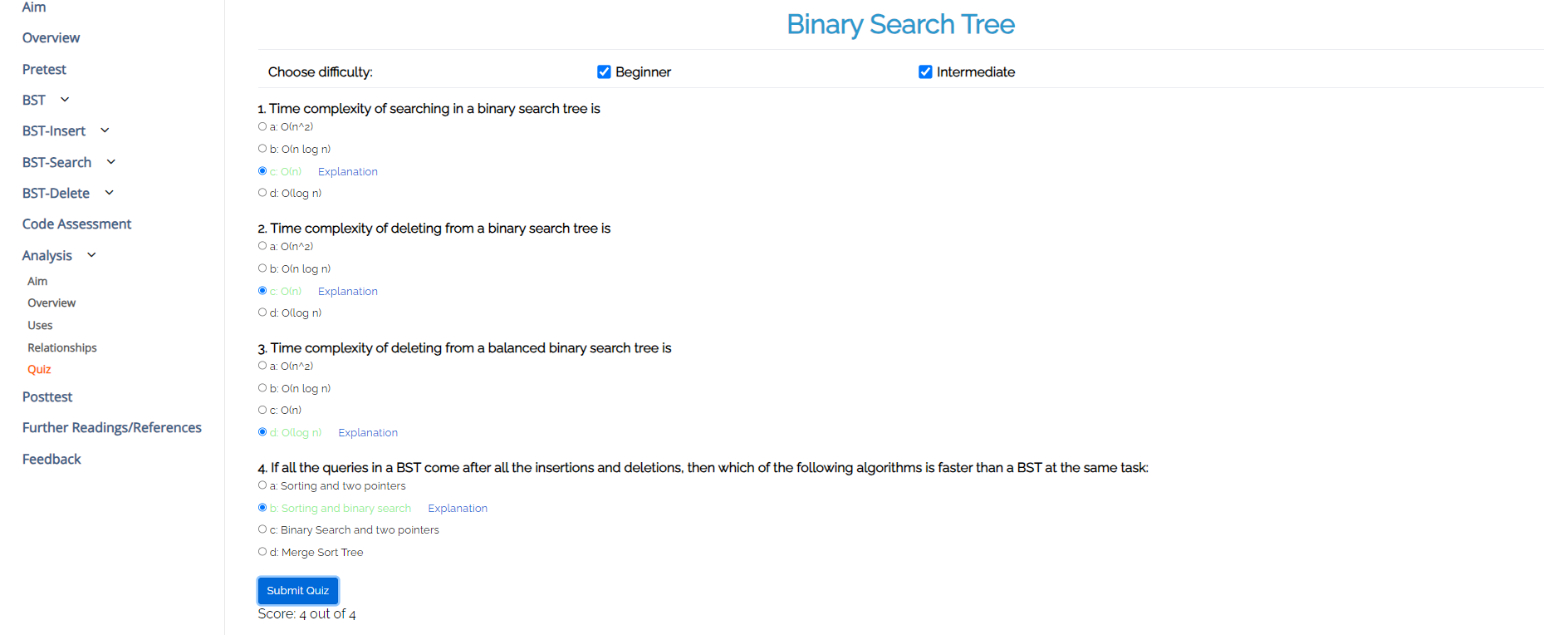
****

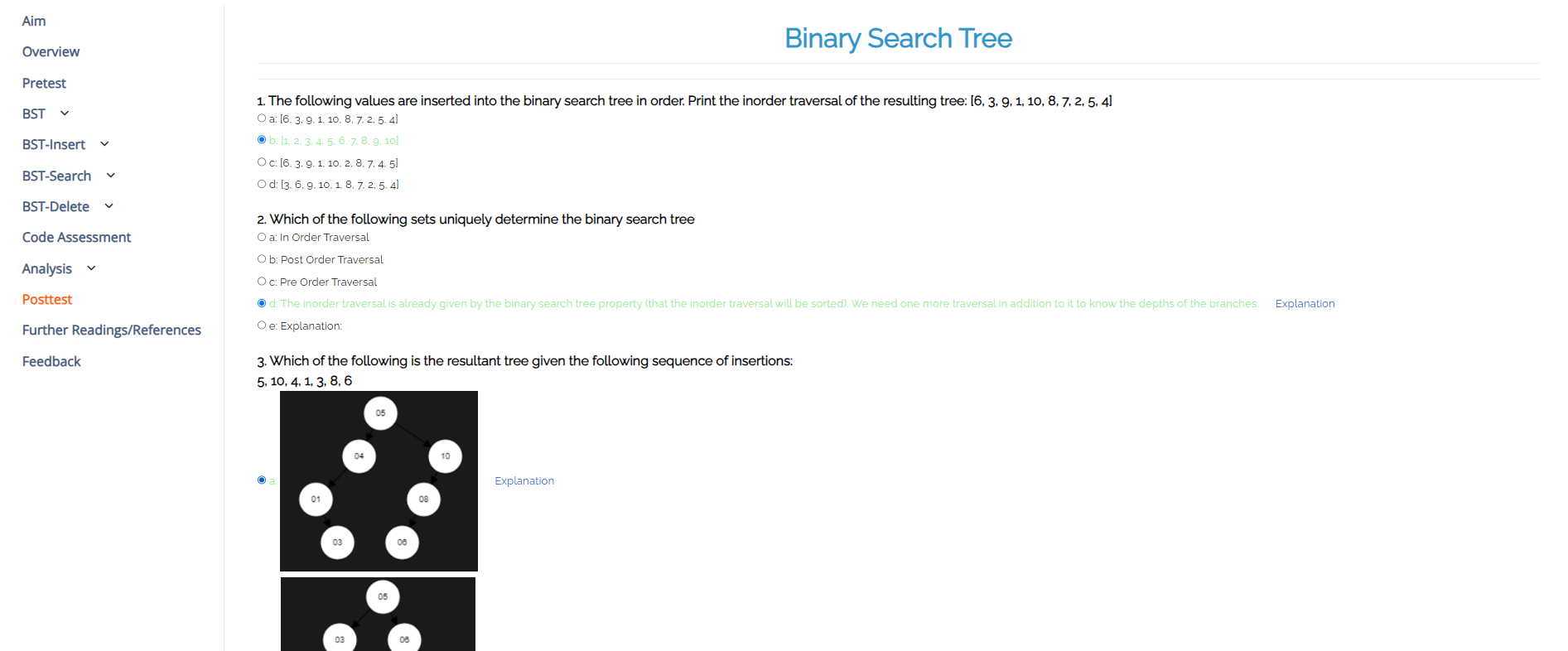
****

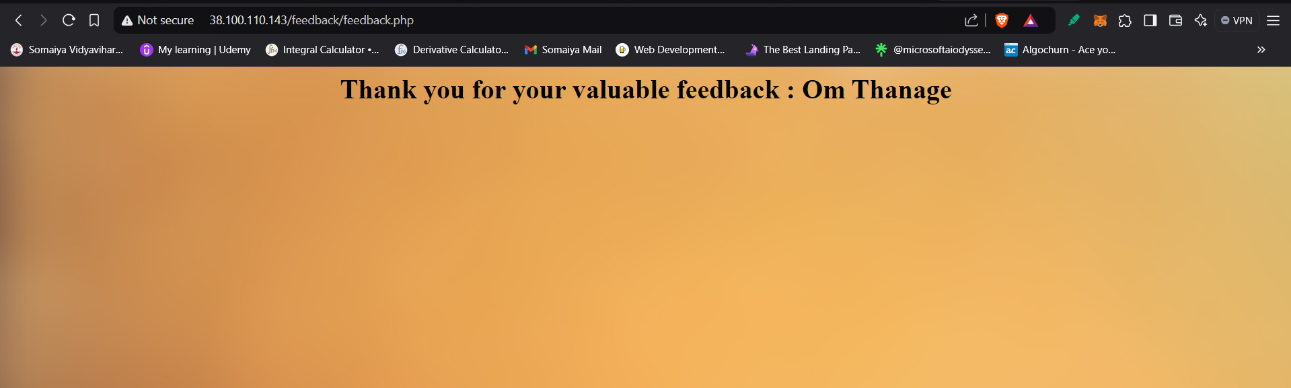
****

****

****

****

****

****