SISTER NIVEDITA UNIVERSITY

School of Engineering

Subject Name:	DATA STRUCTURES AND ALGORITHMS
Subject Code:	
Department:	B.TECH IN COMPUTER SCIENCE AND ENGINEERING

Name:	CHANDREYEE SHOME
Enrolment Number:	2212200001166
Registration Number:	220010415539
Semester:	2nd
Academic Year:	1st (2022 - 2023)

ASSIGNMENT 7

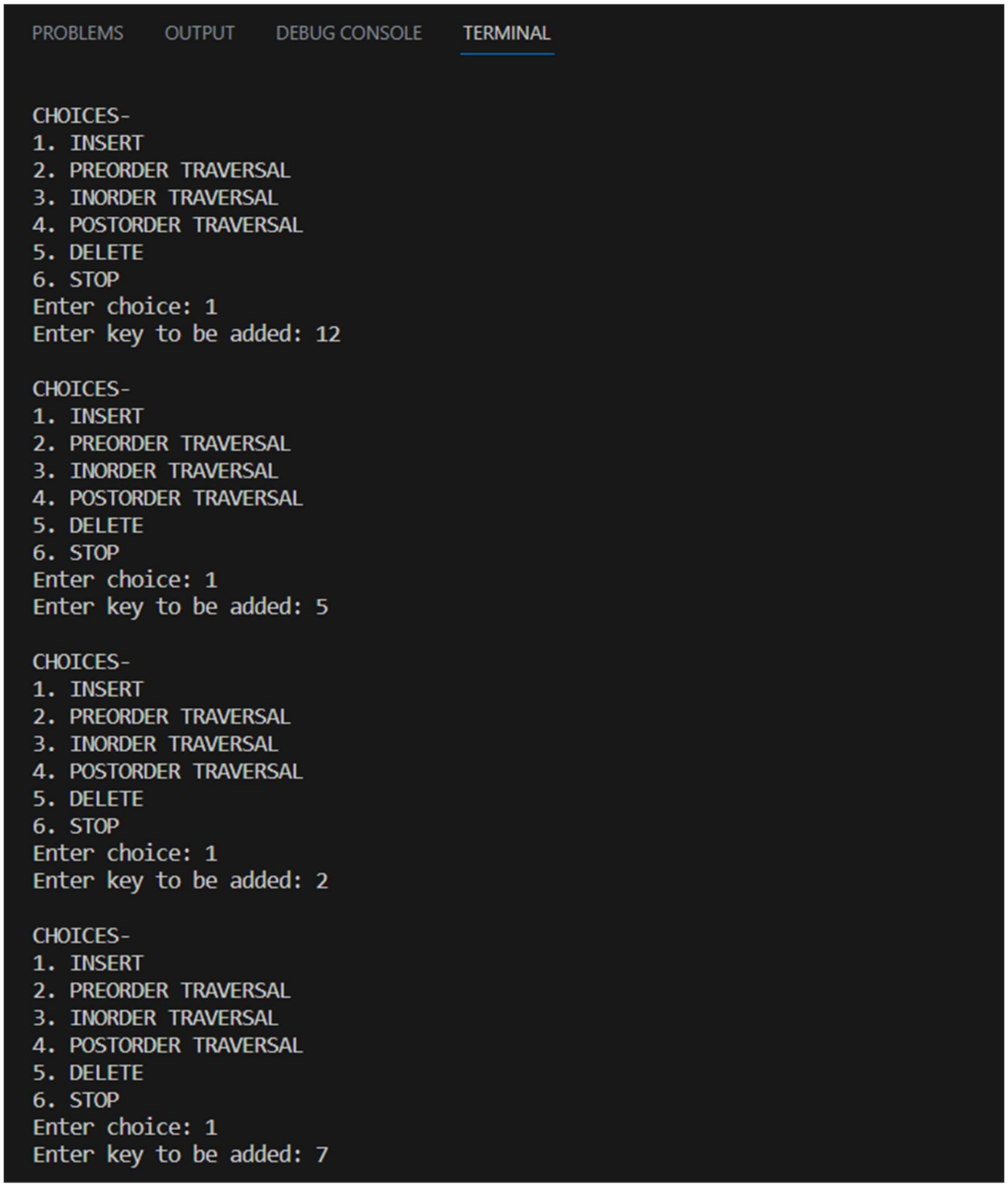
Q 1. Create binary search tree and implement Preorder, Inorder, Postorder and delete an element from the tree.

CODE:

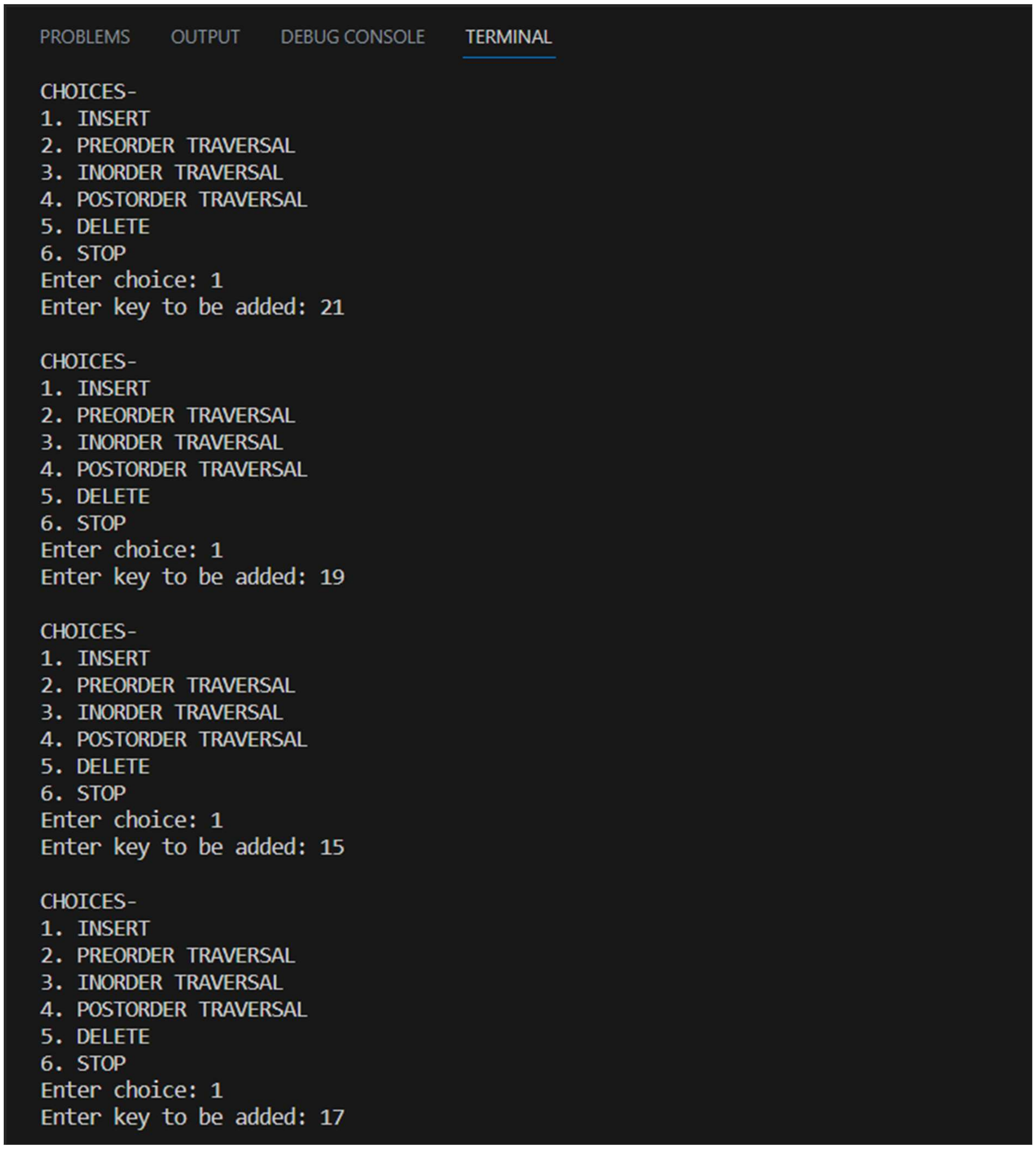
```c
//BINARY SEARCH TREE

//header files
#include<stdio.h>
#include<stdlib.h>

//user defined data structure Node
typedef struct Node
{
    int data;
    struct Node* left;
    struct Node* right;
}Node;

Node* createNode(int data);
Node* insert(Node* root, int data);
void printInorder(Node* node);
void printPreorder(Node* node);
void printPostorder(Node* node);
Node* binarySearch(Node* root, int key);
Node* delete(Node* root, int key);

//function for creating Node of Tree datastructure
Node* createNode(int data)
{
    Node* node;
    node = (Node*)malloc(sizeof(Node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;
    return node;
}

int main()
{
    Node *root = NULL;
    int key, choice;

    system ("cls");
    do
    {
        printf("\nCHOICES-\n1. INSERT\n2. PREORDER TRAVERSAL\n3. INORDER
TRAVERSAL\n4. POSTORDER TRAVERSAL\n5. DELETE\n6. STOP\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("Enter key to be added: ");
                scanf("%d", &key);
                root = insert(root, key);
                break;

            case 2:
                printPreorder(root);
                printf("\n");
                break;

            case 3:
                printInorder(root);
                printf("\n");
                break;

            case 4:
                printPostorder(root);
                printf("\n");
                break;

            case 5:
                printf("Enter key to be deleted: ");
                scanf("%d", &key);
                root = delete(root, key);
                break;

            case 6:
                break;

            default:
                break;
        }
    }while(choice != 6);
}

//function to insert a new node in tree
Node* insert(Node* root, int data)
{
    if(root == NULL)
    {
        root = createNode(data);
    }
    else
    {
        if(data < root->data)
        {
            root->left = insert(root->left, data);
        }
        else
        {
            root->right = insert(root->right, data);
        }
    }
    return root;
}

//function for left node right traversal (LNR)
void printInorder(Node* node)
{
    if (node != NULL)
     {
       printInorder(node->left); // first recur on left child
    printf("%d ",node->data); // then print the data of node
    printInorder(node->right); // now recur on right child
     }

}

//function for node left right traversal (NLR)
void printPreorder(Node* node)
{
    if (node == NULL)
        return;
    printf("%d ",node->data); //first print the data of node
    printPreorder(node->left); //then recur on left child
    printPreorder(node->right); //now recur on right child
}

//function for left right node traversal (LRN)
void printPostorder(Node* node)
{
    if (node == NULL)
        return;
    printPostorder(node->left); //first recur on left child
    printPostorder(node->right); //then recur on right child
    printf("%d ",node->data); //now print data of node
}

//function for binary search
Node* binarySearch(Node* root, int key)
{
    if(root == NULL)
    {
        return NULL;
    }
    else if(root->data == key)
    {
        return root;
    }
    else if(key < root->data)
    {
        return binarySearch(root->left, key);
    }
    else
    {
        return binarySearch(root->right, key);
    }
}

//function to delete a node from tree
Node* delete(Node* root, int key)
{
    struct Node* temp;
    if(root == NULL)
    {
        return root;
    }
    if(key < root->data)
    {
        root->left = delete(root->left, key);
    }
    else if(key > root->data)
    {
        root->right = delete(root->right, key);
    }
    else
    {
        if(root->left == NULL && root->right == NULL)
        { //if leaf node
            free(root);
            return NULL;
        }
        else if(root->left != NULL && root->right == NULL)
        { //node with degree 1 left child
            temp = root;
            root = root->left;
            free(temp);
            return root;
        }
        else if(root->left == NULL && root->right != NULL)
```

```c
        { //node with degree 1 having right child
            temp = root;
            root = root->right;
            free(temp);
            return root;
        }
        else
        { //node with degree 2
            temp = root->right;
            while (temp&&temp->left != NULL)
                temp = temp->left;
            root->data = temp->data;
            root->right = delete(root->right, temp->data);

        }

    }
    return root;
}
```

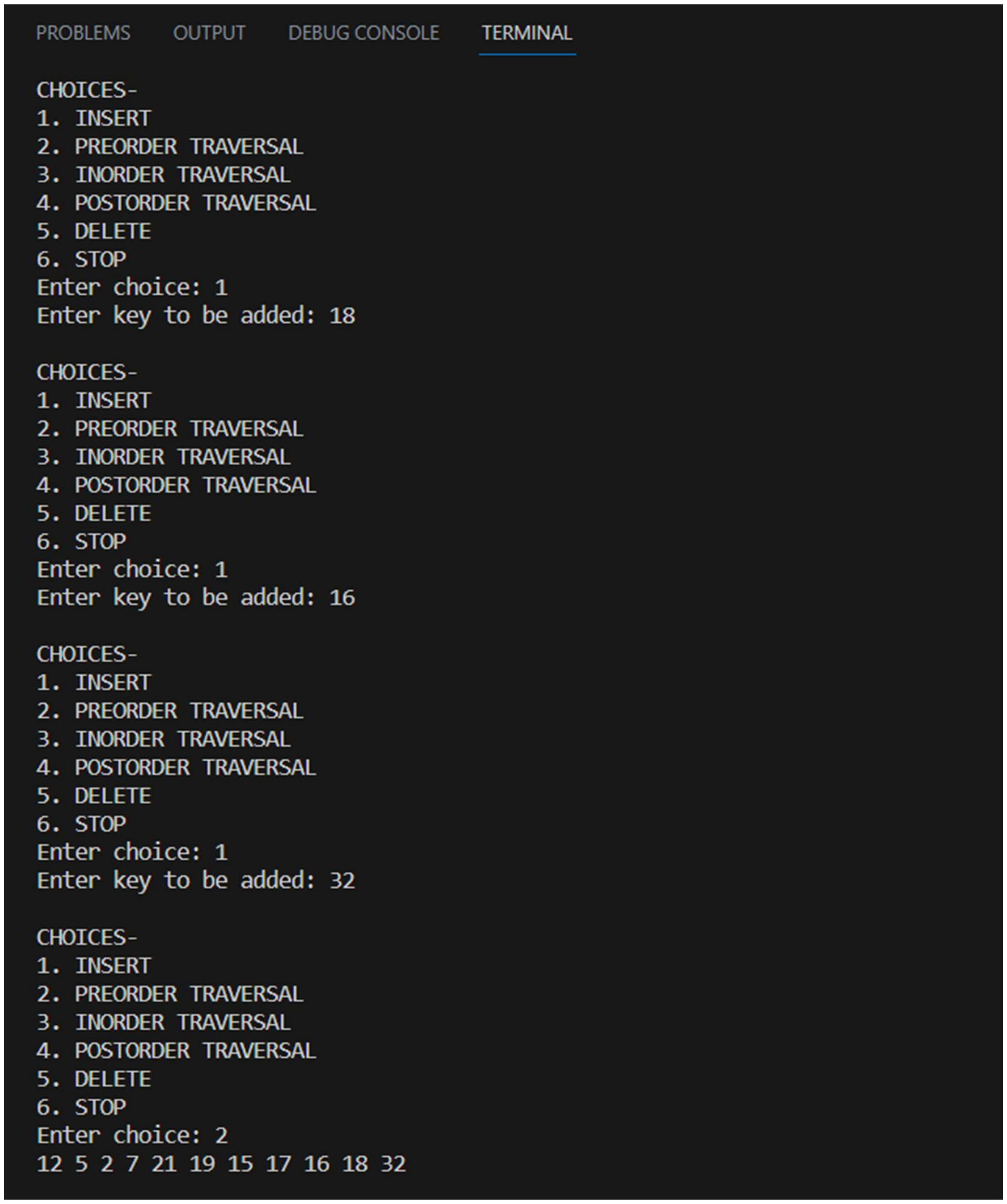
OUTPUT:



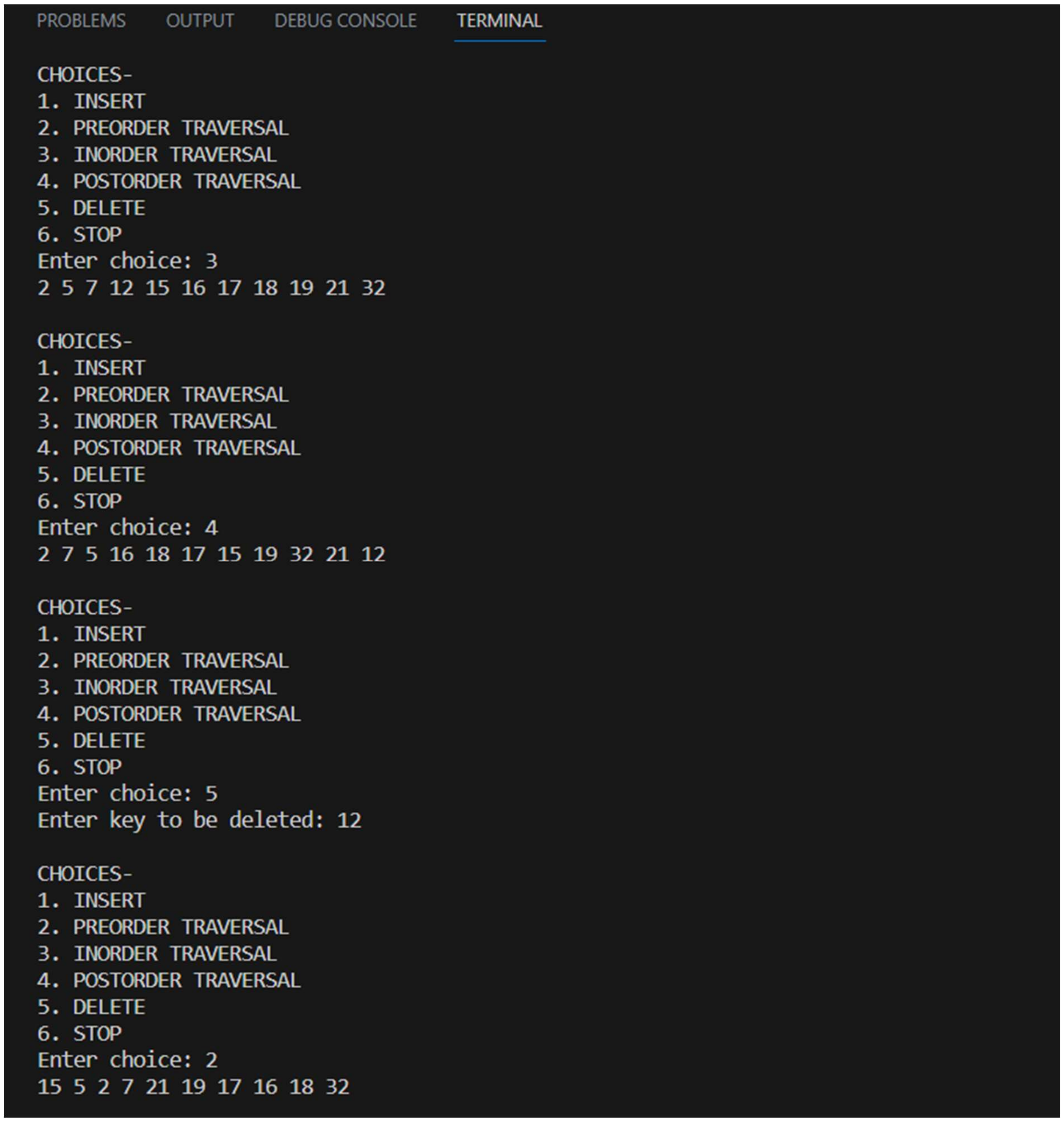
```
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 12

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 5

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 2

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 7
```



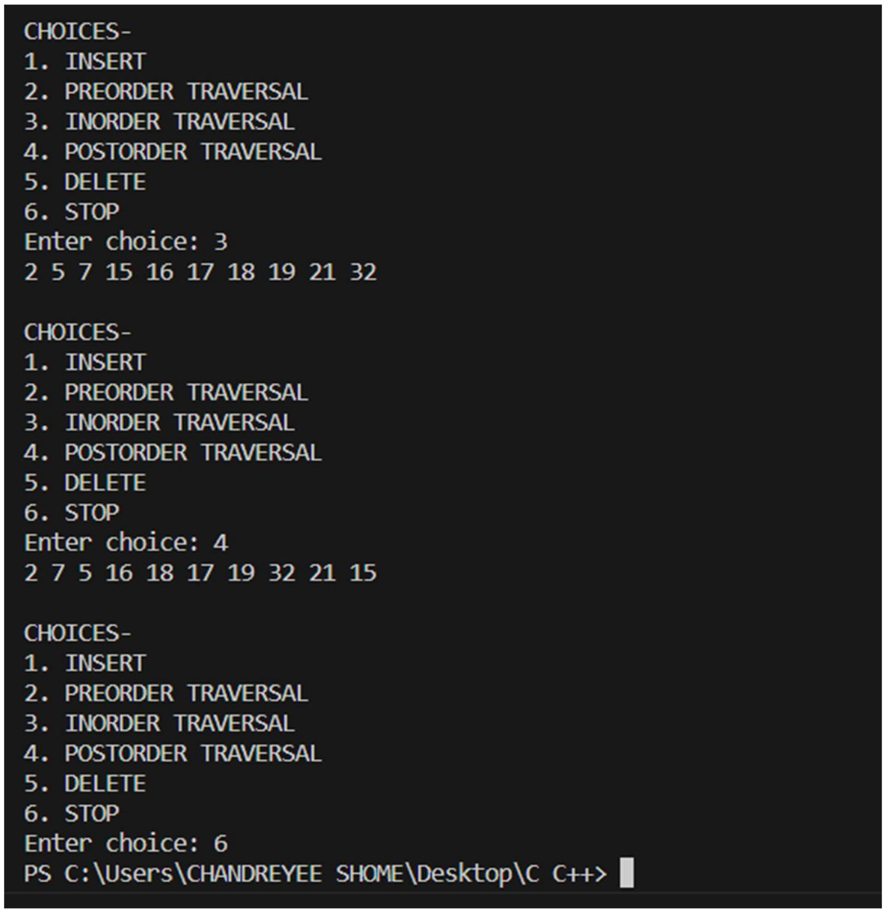
```
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 21

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 19

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 15

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 17
```



```
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 18

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 16

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 1
Enter key to be added: 32

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 2
12 5 2 7 21 19 15 17 16 18 32
```



```
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 3
2 5 7 12 15 16 17 18 19 21 32

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 4
2 7 5 16 18 17 15 19 32 21 12

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 5
Enter key to be deleted: 12

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 2
15 5 2 7 21 19 17 16 18 32
```



```
CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 3
2 5 7 15 16 17 18 19 21 32

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 4
2 7 5 16 18 17 19 32 21 15

CHOICES-
1. INSERT
2. PREORDER TRAVERSAL
3. INORDER TRAVERSAL
4. POSTORDER TRAVERSAL
5. DELETE
6. STOP
Enter choice: 6
PS C:\Users\CHANDREYEE SHOME\Desktop\C C++>
```