Date
Page

Lab - 7.

1. Write a program
a) To construct Binary search tree.
b) Traverse the tree using inorder, postorder, preorder.
c) Display the elements in the tree.

⇒

```
#include<stdio.h>
#include<stdlib.h>

struct BST{
        int data;
        struct BST *Left, *right;
};

struct BST *root = NULL; *temp;

struct BST ~~void~~ create() {
struct BST *temp= malloc (sizeof(struct BST));
        printf ("Enter data: ");
        scanf ("%d", &temp->data);
        temp -> left = temp -> right = NULL;
        return temp;
}
```

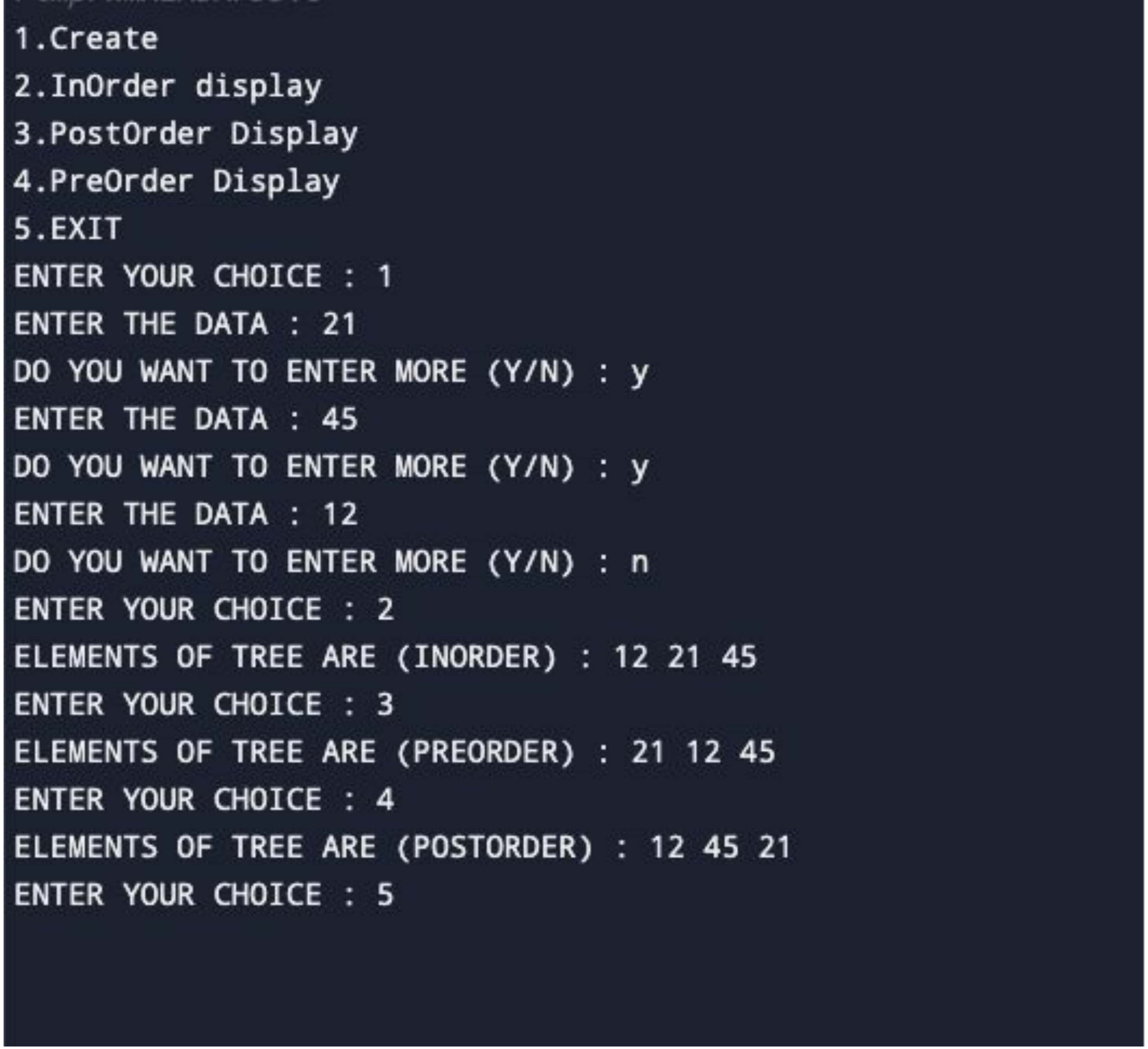
```
void insert(struct BST *root, struct BST *temp)
{
        if (temp->data < root->data){
                if(root->left != NULL)
                        insert(root-> left, temp);
                else
                        root->left = temp;
        }
        else if(temp->data > root->data){
                if(root->right!=NULL)
                        insert(root->right,temp);
                else
                        root->right = temp;
        }
}
```

Executed (1)
[illegible]
19/2/24

```
void preOrder(struct BST *root){
        if(root!=NULL){
                printf("%d", root->data);
                preOrder(root->left);
                preOrder(root->right);
        }
}
```

```
void postOrder (struct BST *root){
        if(root! = NULL){
                postOrder (root -> left);
                postOrder (root -> right);
                printf ("%d", root -> data);
        }
}


void inOrder (struct BST *root){
            if(root! = NULL){
                    inOrder (root-> left);
                    printf("%d", root->data)
                    inOrder (root -> right);
            }
}


~~void~~
int main (){
          int choice;
          char ch;
          struct BST *temp;
          printf (" 1. Create \n 2. Insert \n
                    3. InOrder display \n
                    4. PreOrder display \n
                    5. PostOrder display \n
                    6. Exit \n");
```

```
while(1){
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch(choice){
            case 1: do{
                    temp=create();
                    if(root == NULL)
                            root= temp;
                    else
                            insert(root, temp);
                    printf("Do you want to
                    enter more (Y/N) : ");
                    scanf("%c", &ch);
            }while(ch=='Y' || ch=='y');
            break;

          case 2:
                printf("Elements of tree
                 (InOrder) : ");
                inOrder(root);
                break;

          case 3:
                printf("Elements of tree
                 (PreOrder): ");
                preOrder(root);
                break;
            case 4:
                printf("Elements of tre
                 (postOrder ): ");
                postOrder(root);
                break;

            case 5:
                    exit(0);

            default:
                    printf("Invalid inputs:

        }
    }
    return 0;
}
```



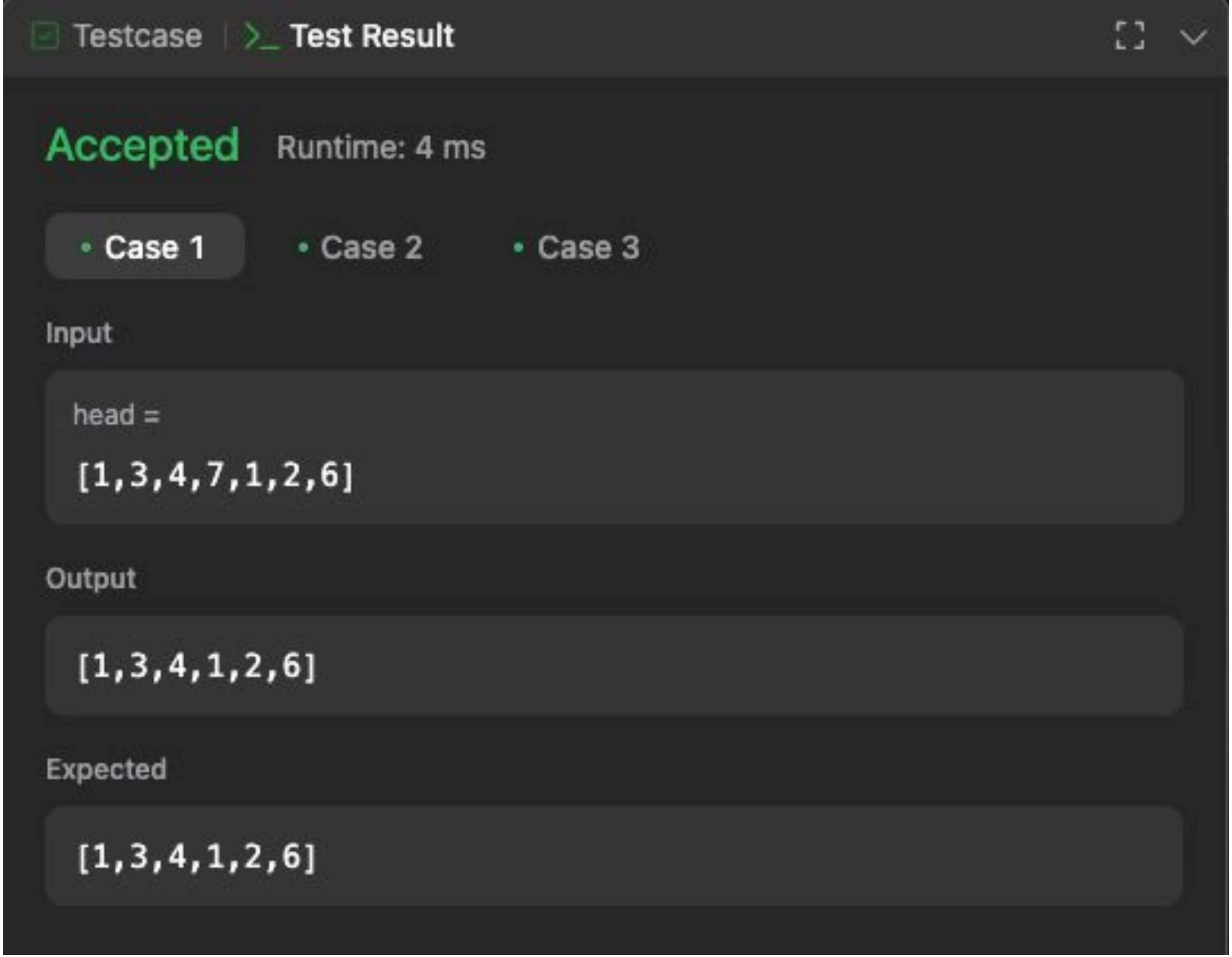
```
1.Create
2.InOrder display
3.PostOrder Display
4.PreOrder Display
5.EXIT
ENTER YOUR CHOICE : 1
ENTER THE DATA : 21
DO YOU WANT TO ENTER MORE (Y/N) : y
ENTER THE DATA : 45
DO YOU WANT TO ENTER MORE (Y/N) : y
ENTER THE DATA : 12
DO YOU WANT TO ENTER MORE (Y/N) : n
ENTER YOUR CHOICE : 2
ELEMENTS OF TREE ARE (INORDER) : 12 21 45
ENTER YOUR CHOICE : 3
ELEMENTS OF TREE ARE (PREORDER) : 21 12 45
ENTER YOUR CHOICE : 4
ELEMENTS OF TREE ARE (POSTORDER) : 12 45 21
ENTER YOUR CHOICE : 5
```

* Delete the Middle Node.

```
struct ListNode *deleteMiddle(struct
  ListNode * head){
        if(!head || !head-> next){
             return NULL;
        }
        int count=0

        struct listNode * curr=head;
        while(curr) {
              count++
              curr=curr->next;
        }
        int mid = count/2;
        curr=head;

         if(mid ==0){
              head=head->next
              free(curr);
               return head;
         }.

         for(int i=0; i<mid-1; i++)
              curr=curr->next;
```

```
    struct ListNode *temp=curr->n
    curr-> next =curr->next->nex
    free (temp)
     return (head);
}
```

[illegible] 26/2/24



Accepted Runtime: 4 ms

• Case 1 • Case 2 • Case 3

Input

```
head =
[1,3,4,7,1,2,6]
```

Output

```
[1,3,4,1,2,6]
```

Expected

```
[1,3,4,1,2,6]
```