NAME :	Maryada Lodha
EXPERIMENT NO :	05
DATE :	21-11-2022

AIM :	Insertion in Binary Search Tree

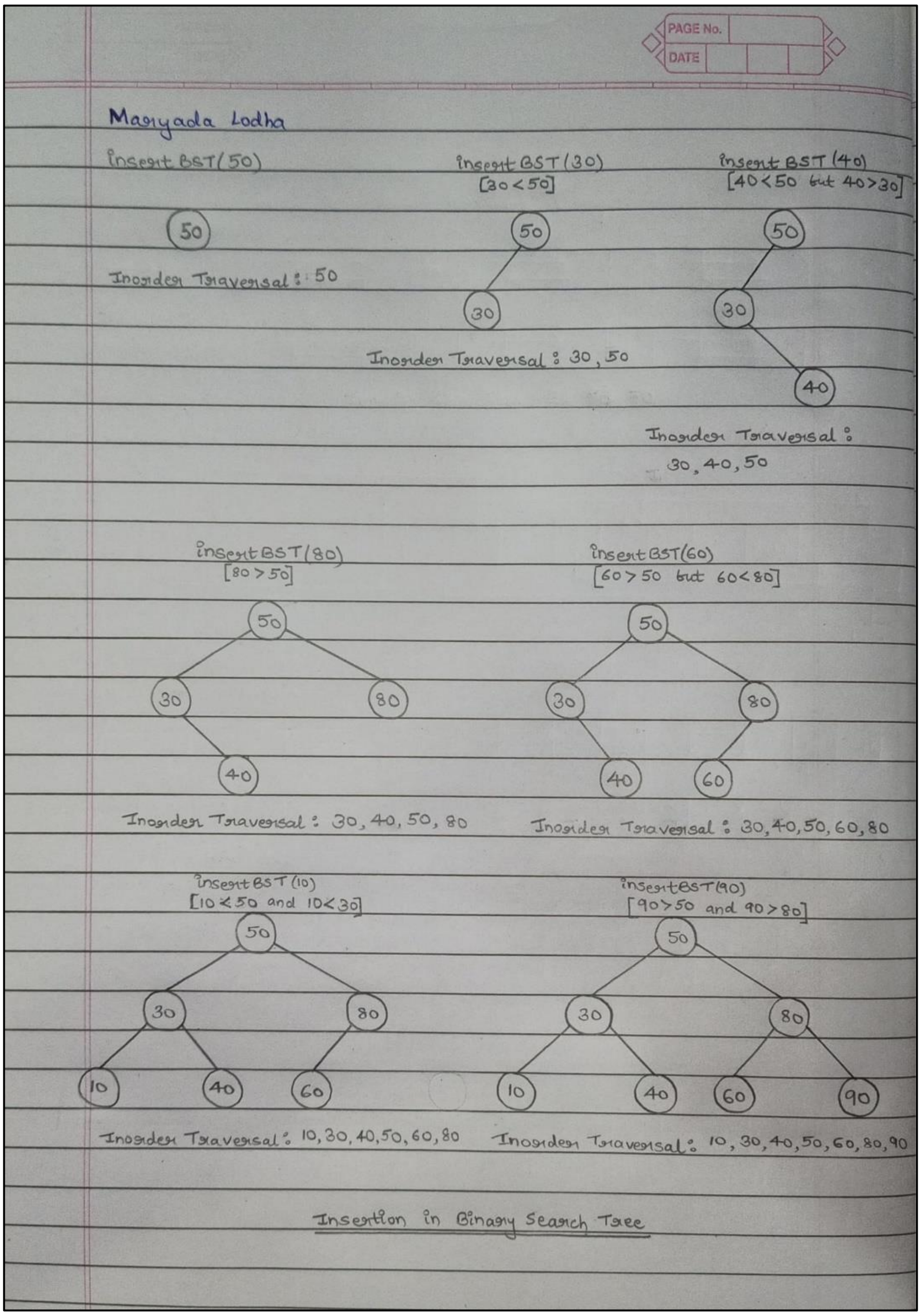
ALGORITHM :

```
int main()

1] Declare ROOT node and assign NULL value to it

2] Repeat steps in infinite loop

3] Read user’s choice,

          i] To insert a new node

         ii] To Display In-Order Traversal of Tree.

        iii] To Display Pre-Order Traversal of Tree

        iv] To Display Post-Order Traversal of Tree

4] Switch Case 1 :

         i] Read DATA value of the new node to be inserted

        ii] Call insertBST function with the ROOT node and DATA

             value and assign the return value to the ROOT node

5] Switch Case 2 :

        i] Call inorderTraversal function with ROOT node

6] Switch Case 3:

        i] Call preorderTraversal function with ROOT node

7] Switch Case 4:

        i] Call postorderTraversal function with ROOT node

8] Return 0
```

```
struct node * create(int a)
1] Allocate memory for Newnode
2] Set DATA of the Newnode as a
3] Set LEFT pointer and RIGHT pointer of Newnode to NULL
4] Return Newnode

struct node * insertBST(struct node *root, int a)
1] If ROOT = = NULL
         Create a Newnode and return the pointer to that node
   Else If value of a < ROOT -->DATA
         Call the insertBST function with ROOT -->LEFT and data value a
         and assign the return value to ROOT -->LEFT
   Else If value of a > ROOT -->DATA
         Call the insertBST function with ROOT -->RIGHT and data value a
         and assign the return value to ROOT -->RIGHT
   [End If]
2] Print “Node Inserted”
3] Return the pointer to the original ROOT to the calling function;
4] End

void inorderTraversal(struct node *root)
1] If ROOT = = NULL
       Return to the calling function
2] Call the inoderTraversal function with ROOT -->LEFT to traverse the
   left subtree
```

```
3] Visit the ROOT node and print ROOT -->DATA
4] Call the inoderTraversal function with ROOT -->RIGHT to traverse the
   right subtree
5] End

void preorderTraversal(struct node *root)
1] If ROOT = = NULL
        Return to the calling function
2] Visit the ROOT node and print ROOT -->DATA
3] Call the preoderTraversal function with ROOT -->LEFT to traverse the
   left subtree
4] Call the preoderTraversal function with ROOT -->RIGHT to traverse the
   right subtree
5] End

void postorderTraversal(struct node *root)
1] If ROOT = = NULL
        Return to the calling function
2] Call the preoderTraversal function with ROOT -->LEFT to traverse the
   left subtree
3] Call the preoderTraversal function with ROOT -->RIGHT to traverse the
   right subtree
4] Visit the ROOT node and print ROOT -->DATA
5] End
```

PROBLEM SOLVING :



PAGE No.

DATE

Maryada Lodha

insert BST(50)

Inorder Traversal : 50

insert BST(30)
[30<50]

Inorder Traversal : 30, 50

insert BST (40)
[40<50 but 40>30]

Inorder Traversal :
30, 40, 50

insert BST(80)
[80>50]

Inorder Traversal : 30, 40, 50, 80

insert BST(60)
[60>50 but 60<80]

Inorder Traversal : 30,40,50,60,80

insert BST (10)
[10<50 and 10<30]

Inorder Traversal : 10,30,40,50,60,80

insertBST(90)
[90>50 and 90>80]

Inorder Traversal : 10, 30,40,50,60,80,90

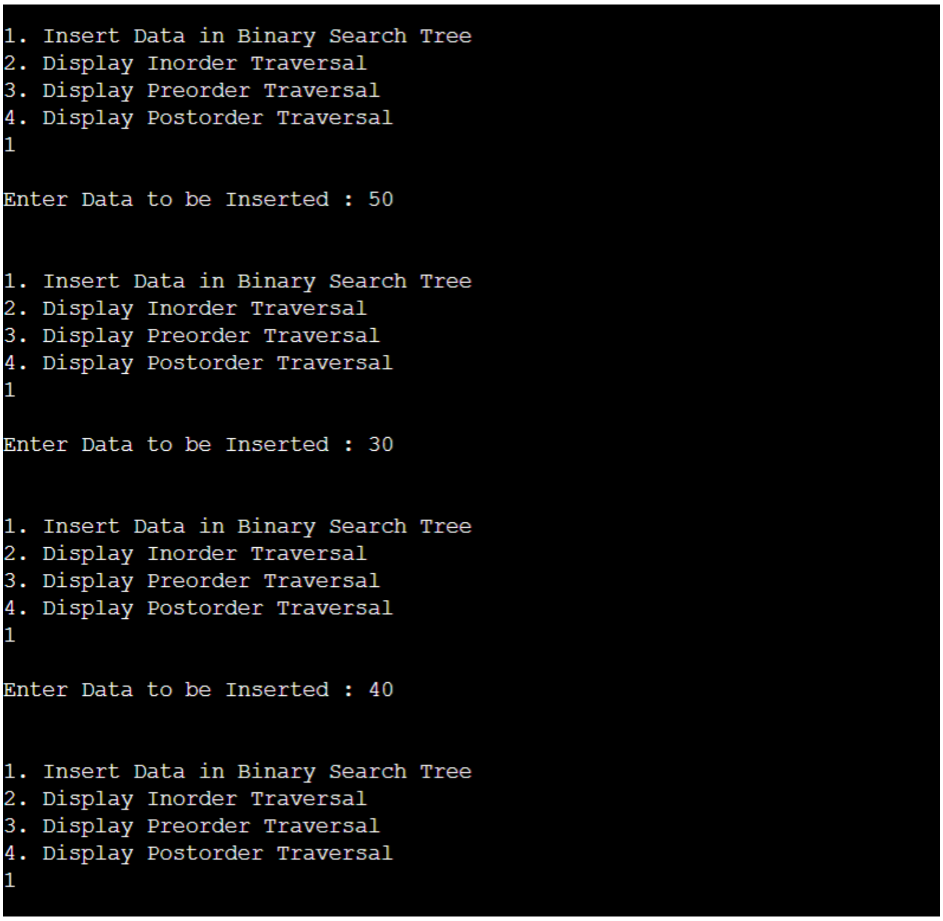
Insertion in Binary Search Tree

CODE :

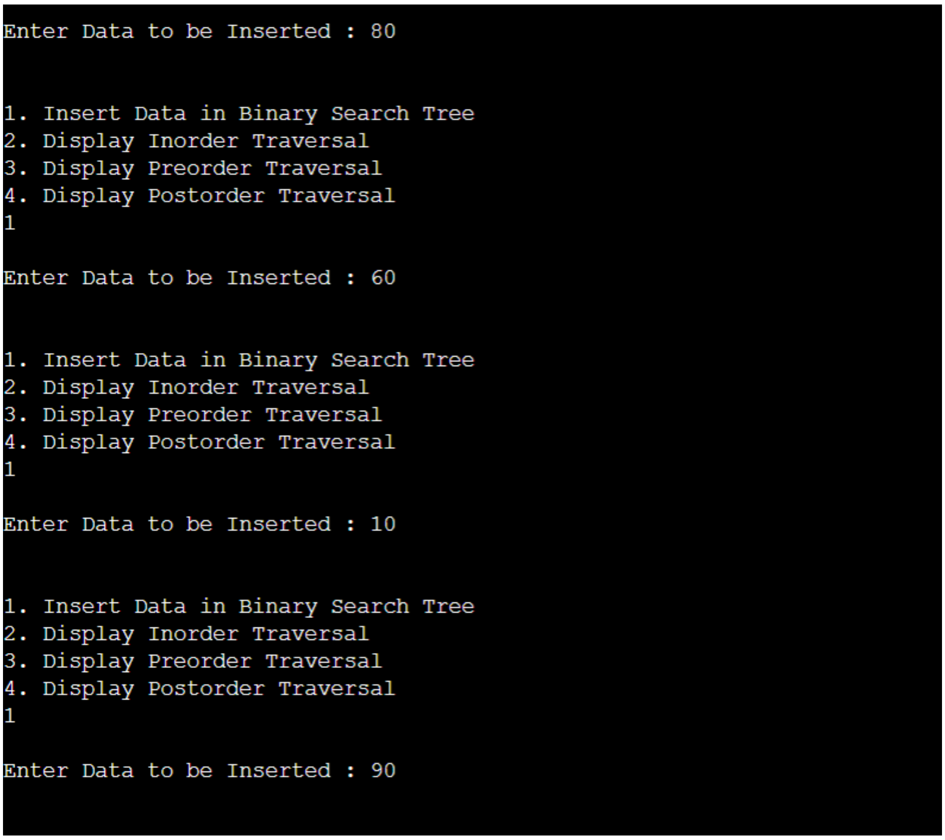
```
#include <stdio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

struct node * create(int a);
struct node * insertBST(struct node *root,int a);
void inorderTraversal(struct node *root);
void preoderTraversal(struct node *root);
void postorderTraversal(struct node *root);

int main()
{
    int choice,a,i=7;
    struct node *root=NULL;

    while(1)
    {
        printf("\n\n1. Insert Data in Binary Search Tree\n2. Display Inorder Traversal\n3. Display Preorder Traversal\n4. Display Postorder Traversal\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("\nEnter Data to be Inserted : ");
                scanf("%d",&a);
                root=insertBST(root,a);
                break;
            case 2:
                printf("\nInorder Traversal of Binary Search Tree : ");
                inorderTraversal(root);
                break;
            case 3:
                printf("\nPreorder Traversal of Binary Search Tree : ");
                preoderTraversal(root);
                break;
            case 4:
                printf("\nPostorder Traversal of Binary Search Tree : ");
                postorderTraversal(root);
                break;
        }
    }
    return 0;
}

struct node * create(int a)
{
    struct node *newnode=(struct node *) malloc(sizeof(struct node));
    newnode->data=a;
    newnode->left=NULL;
    newnode->right=NULL;
    return newnode;
}

struct node * insertBST(struct node *root,int a)
{
    if(root==NULL)
    {
        return create(a);
    }
    else if(a<root->data)
    {
        root->left=insertBST(root->left,a);
    }
    else if(a>root->data)
    {
        root->right=insertBST(root->right,a);
    }
    return root;
}
```

```
void inorderTraversal(struct node * root)
{
   if(root==NULL)
   {
      return;
   }
   inorderTraversal(root->left);
   printf("%d, ",root->data);
   inorderTraversal(root->right);
}

void preoderTraversal(struct node *root)
{
   if(root==NULL)
   {
      return;
   }
   printf("%d, ",root->data);
   preoderTraversal(root->left);
   preoderTraversal(root->right);
}

void postorderTraversal(struct node *root)
{
   if(root==NULL)
   {
      return;
   }
   postorderTraversal(root->left);
   postorderTraversal(root->right);
   printf("%d, ",root->data);
}
```

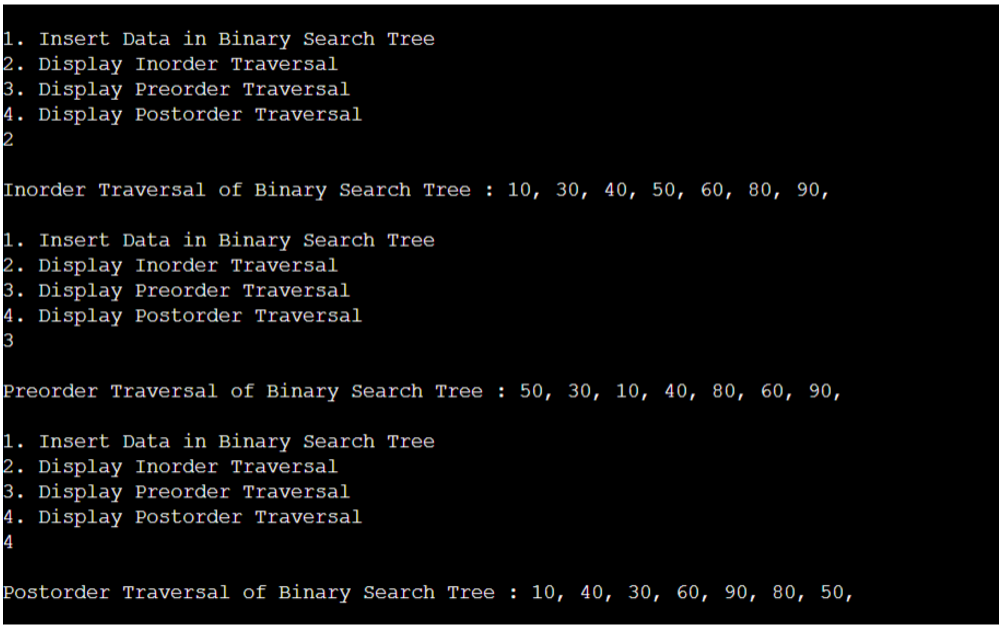
OUTPUT :



```
1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1

Enter Data to be Inserted : 50


1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1

Enter Data to be Inserted : 30


1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1

Enter Data to be Inserted : 40


1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1
```



```
Enter Data to be Inserted : 80


1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1

Enter Data to be Inserted : 60


1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1

Enter Data to be Inserted : 10


1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
1

Enter Data to be Inserted : 90
```



```
1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
2

Inorder Traversal of Binary Search Tree : 10, 30, 40, 50, 60, 80, 90,

1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
3

Preorder Traversal of Binary Search Tree : 50, 30, 10, 40, 80, 60, 90,

1. Insert Data in Binary Search Tree
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
4

Postorder Traversal of Binary Search Tree : 10, 40, 30, 60, 90, 80, 50,
```