Name:- Brendan Lucas , Roll No:- 8953

Binary Tree implication Code:-

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
 {
        int data;
        struct node *left,*right;
 }node;
 typedef struct
 {
node *start;
 }bt;

  void insert(bt* start)
 {
        node *p,*q,*r;
        int i;
        p=(node*)malloc(sizeof(node ));
        printf("Enter The Data To be Inserted in the tree\n");
        scanf("%d",&i);
        p->data=i;
        //printf("%d\n",p->data);
        p->left=p->right=NULL;
        q=start->start;
        if(q==NULL)
        {
                //printf("1\n");
                start->start=p;
        }
        else
        {
                q=start->start;
                while(q)
                {
                        r=q;
                        if(p->data<q->data)
                        {
                                q=q->left;
                                //printf("left\n");
                        }
                        else if(p->data>=q->data)
                        {
                                q=q->right;
                                //printf("right\n");
                        }
                }
                if(p->data>=r->data)
                {
                        r->right=p;
```

```
                        //printf("rightson\n");
                }
                else
                {
                        r->left=p;
                        //printf("leftson\n");
                }
        }
        printf("Node Successfully Inserted in the Tree\n");
}
void preorder(node* n)
{
        node *p=n;
        if(p==NULL)
        {
                printf("Tree is Empty\n");
                return;
        }
        printf("%d  ",p->data);
        if(p->left)
        {
                preorder(p->left);
        }
        if(p->right)
        {
                preorder(p->right);
        }
}
void inorder(node* n)
{
        node *p=n;
        if(p==NULL)
        {
                printf("Tree is Empty\n");
                return;
        }
        if(p->left)
        {
                inorder(p->left);
        }
        printf("%d  ",p->data);
        if(p->right)
        {
                inorder(p->right);
        }
}
void postorder(node* n)
{
        node *p=n;
        if(p==NULL)
        {
                printf("Tree is Empty\n");
                return;
```

```
        }
        if(p->left)
        {
                postorder(p->left);
        }
        if(p->right)
        {
                postorder(p->right);
        }
        printf("%d  ",p->data);
}
/*node**/void search_itr(bt* start/*,int k*/)
{
        int k,f=0;
        node *p,*q;
        printf("Enter the Number to searched\n");
        scanf("%d",&k);
        q=start->start;
        while(q!=NULL)
        {
                if(q->data==k)
                {
                        f=1;
                        break;
                }
                else if(k>q->data)
                 {
                        q=q->right;
                 }
                 else
                 {
                        q=q->left;
                 }
        }
        if(f==1)
        {
                printf("%d has been found\n",q->data);
                //return q;
        }
        else
        {
                printf("%d is not found\n",k);
                //return NULL;
        }
}

node* search_rec(node *p,int k)
{
        if(p==NULL)
        {
                return   NULL;
        }
        if(p->data==k)
```

```
        {
                //printf("%d has been found",p->data);
                return p;
        }
        else if(k>p->data)
        {
                return search_rec(p->right,k); ;
        }
        else
        {
                return search_rec(p->left, k);
        }
}
int countNodes(node *p)
{
        if(!p)
        {
                return 0;
        }
        else
        {
                return (1+countNodes(p->left)+countNodes(p->right));
        }
}
int countleaves(node *p)
{
        if(!p)
        {
                return 0;
        }
        else if(!p->left&&!p->right)
        {
                return 1;
        }
        else
        {
                return (countleaves(p->left)+countleaves(p->right));
        }
}

/*node**/ void maxInTree(bt* start)
{
        node *p=start->start;
        if(!p)
        {
                printf("Tree is empty\n");
                return /*NULL*/;
        }
        while(p->right)
        {
                p=p->right;
        }
        //return p;
```

```
	printf("The Largest Number in the tree is %d\n",p->data);
}
/*node**/ void minInTree(bt* start)
{
	node *p=start->start;
	if(!p)
	{
		printf("Tree is empty\n");
		return /*NULL*/;
	}
	while(p->left)
	{
		p=p->left;
	}
	//return p;
	printf("The Smallest Number in the tree is %d\n",p->data);
}

int heightOfTree(node *p)
{
	int i,j;
	if(!p)
	{
		return -1;
	}
	if(p->right==NULL&&p->left==NULL)
	{
		return 0;
	}
	else
	{
		i=heightOfTree(p->left);
		j=heightOfTree(p->right);
		return (i>j)?(i+1):(j+1);
	}
}
void mirrorTree(node* p)
{
	node* temp;
	if(!p)
	{
		return;
	}
	else
	{
		temp=p->left;
		p->left=p->right;
		p->right=temp;
		mirrorTree(p->left);
		mirrorTree(p->right);
	}
}
void deletenode(bt* start/*,int x*/)
```

```
{
	node *p,*q,*r;
	int x;
	printf("Enter the data to be deleted:-\n");
	scanf("%d",&x);
	r=NULL;
	p=start->start;
	while(p!=NULL)
	{
		if(x==p->data)
		{
			break;
		}
		r=p;
		if(x>(p->data))
		{
			p=p->right;
		}
		else
		{
			p=p->left;
		}
	}
	if(!p)
	{
		printf("Data Not found in tree\n");
		return;
	}
	if(!p->left&&!p->right)
	{
		if(r->left==p)
		{
			r->left=NULL;
		}
		else
		{
			r->right=NULL;
		}
		printf("%d has been deleted from the tree\n",p->data);
		free(p);
		return;
	}
	if(!p->left)
	{
		if(r->left==p)
		{
			r->left=p->right;
		}
		else
		{
			r->right=p->right;
		}
		printf("%d has been deleted from the tree\n",p->data);
```

```
                free(p);
                return;
        }
        if(!p->right)
        {
                if(r->left==p)
                {
                        r->left=p->left;
                }
                else
                {
                        r->right=p->left;
                }
                printf("%d has been deleted from the tree\n",p->data);
                free(p);
                return;
        }
        else
        {
                r=q=p;
                q=q->right;
                while(q->left)
                {
                        r=q;
                        q=q->left;
                }
                p->data=q->data;
                if(r==p)
                {
                        p->right=q->right;
                }
                else if(q->right!=NULL)
                {
                        r->left=q->right;
                }
                else
                {
                        r->left=NULL;
                }
                printf("%d has been deleted from the tree\n",x);
                free(q);
                return;
        }
 }
int main()
{
        bt tree1;
        char ch='y';
        tree1.start=NULL;
        int c;
        node *j;
        while(ch=='y'||ch=='Y')
        {
```

```
		printf("\nEnter Your Choice\n1 for Insertion\n2 for preorder\n3 for inorder\n4 for postorder\n5 for search by iterative\n6 for search by recursion\n7 for Counting nodes\n8 for counting leaves\n9 for minimum in tree\n10 for maximum in tree\n11 for height of tree\n12 for mirror of tree\n13 for deleting an element\n0 for exit\n\n");
		scanf("%d",&c);
		switch(c)
		{case 1:{insert(&tree1);break;}
			case 2:{preorder(tree1.start);printf("\n");break;}
			case 3:{inorder(tree1.start);printf("\n");break;}
			case 4:{postorder(tree1.start);printf("\n");break;}
			case 5:{search_itr(&tree1);break;}
			case 6:{
				printf("Enter the number to be searched\n");
				scanf("%d",&c);
				j=search_rec(tree1.start, c);
				if(j==NULL)
				{
					printf("%d has not been found\n",c);
				}
				else
				{
					printf("%d has been found\n",j->data);
				}
				break;
				}
			case 7:{
				c=countNodes(tree1.start);
				printf("The total nodes are %d\n",c);
				break;
				}
			case 8:{
				c=countleaves(tree1.start);
				printf("The total Leaves nodes are %d\n",c);
				break;
				}
			case 9:{minInTree(&tree1);break;}
			case 10:{maxInTree(&tree1);break;}
			case 11:{
				c=heightOfTree(tree1.start);
				printf("The Height of tree is %d\n",c);
				break;
				}
			case 12:{mirrorTree(tree1.start);printf("Tree has been mirrored\n");break;}
			case 13:{deletenode(&tree1);break;}
			case 0:exit(0);
			default:{
				printf("Invalid option. Please enter a valid input\n\n");break;
				}
		}
		printf("Do You Want to Continue,If Yes press Y :-\n");
		scanf("%c",&ch);
	}
	return 0;
}
```

Output:-

```
Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
5
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
4
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
```

2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
6
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
2
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves

9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
1
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
3
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
8
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
7
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

1
Enter The Data To be Inserted in the tree
9
Node Successfully Inserted in the Tree
Do You Want to Continue,If Yes press Y :-
y

```
Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

3
1  2  3  4  5  6  7  8  9
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

2
5  4  2  1  3  6  8  7  9
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
```

11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

4
1 3 2 4 7 9 8 6 5
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

5
Enter the Number to searched
8
8 has been found
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

6
Enter the number to be searched
4
4 has been found

Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

7
The total nodes are 9
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

8
The total Leaves nodes are 4
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes

8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

9
The Smallest Number in the tree is 1
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

10
The Largest Number in the tree is 9
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

11
The Height of tree is 3
Do You Want to Continue,If Yes press Y :-

y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

13
Enter the data to be deleted:-
8
8 has been deleted from the tree
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion
7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

12
Tree has been mirrored
Do You Want to Continue,If Yes press Y :-
y

Enter Your Choice
1 for Insertion
2 for preorder
3 for inorder
4 for postorder
5 for search by iterative
6 for search by recursion

7 for Counting nodes
8 for counting leaves
9 for minimum in tree
10 for maximum in tree
11 for height of tree
12 for mirror of tree
13 for deleting an element
0 for exit

3
9 7 6 5 4 3 2 1
Do You Want to Continue,If Yes press Y :-
n