

```

//Binary Search tree
#include<stdio.h>
#include<stdlib.h>
struct Tnode{
    int e;
    struct Tnode *left;
    struct Tnode *right;
}*temp,*templ,*parent;
struct Tnode *root=NULL;
struct Queue
{
    struct Tnode *data[100];
    int rear;
    int front;
    int max;
};
struct Tnode *insert(struct Tnode *root);
struct Tnode *find(struct Tnode *root);
struct Tnode *delete(struct Tnode *root);
void enqueue(struct Queue *Q,struct Tnode *root)
{
    if((Q->rear+1)==Q->max)
        printf("Queue is full\n");
    else
    {
        (Q->rear)++;
        Q->data[Q->rear]=root;
    }
}
struct Tnode* dequeue(struct Queue *Q)
{
    if(Q->front==Q->rear)
        printf("Queue is empty\n");
    else
    {
        return Q->data[++Q->front];
    }
}
int isempty(struct Queue *Q)
{
    if(Q->rear==Q->front)
        return 0;
    else
        return 1;
}
struct Tnode *insert(struct Tnode *root)
{
    int ele;
    printf("\nEnter Element that u want to insert\n");
    scanf("%d",&ele);
    if(root==NULL){
        root=(struct Tnode*)malloc(sizeof(struct Tnode));
        root->e=ele;
        root->left=root->right=NULL;
    }
    else{
        struct Tnode *temp=root;
        while(1)
        {
            if(ele>temp->e)
            {
                if(temp->right!=NULL)
                    temp=temp->right;
                else{
                    struct Tnode *templ;
                    templ=(struct Tnode*)malloc(sizeof(struct Tnode));
                    templ->e=ele;
                    templ->left=temp->right=NULL;
                    temp->right=templ;
                    break;
                }
            }
        }
    }
}

```

```

        else{
            if(temp->left!=NULL)
                temp=temp->left;
            else{
                struct Tnode *temp1;
                temp1=(struct Tnode*)malloc(sizeof(struct Tnode));
                temp1->e=ele;
                temp1->right=temp1->left=NULL;
                temp->left=temp1;
                break;
            }//else
        }//while
    }//else
    return root;
}

struct Tnode *find(struct Tnode *root)
{
    int ele;
    printf("Enter Element to Search :");
    scanf("%d",&ele);
    if(root==NULL){
        printf("\tTree is Empty\n");
        return NULL;
    }
    else{
        struct Tnode *temp=root;
        while(1)
        {
            if(ele==temp->e){
                printf("\tElement is Found\n");
                return temp;
            }
            else if(ele>temp->e){
                if(temp->right!=NULL)
                    temp=temp->right;
                else{
                    printf("\tElement Not Found\n");
                    return NULL;
                }
            }
            else{
                if(temp->left!=NULL)
                    temp=temp->left;
                else{
                    printf("\tElement is Not Found\n");
                    return NULL;
                }
            }//else
        }//while
    }//else
}

struct Tnode *findtemp(struct Tnode *root,int ele)
{
    if(root==NULL){
        printf("\tTree is Empty\n");
        return NULL;
    }
    else{
        struct Tnode *temp=root;
        while(1)
        {
            if(ele==temp->e)
                return temp;
            else if(ele>temp->e){
                if(temp->right!=NULL)
                    temp=temp->right;
                else
                    return NULL;
            }
            else{
                if(temp->left!=NULL)
                    temp=temp->left;
                else
                    return NULL;
            }
        }
    }
}

```

```

        if(temp->left!=NULL)
            temp=temp->left;
        else
            return NULL;
    }//else
    }//while
}
struct Tnode *findparent(struct Tnode *root,struct Tnode *temp)
{
    while(1)
    {
        if(root->left==temp)
            return root;
        else if(root->right==temp)
            return root;
        else{
            if(temp->e<root->e)
                root=root->left;
            else
                root=root->right;
        }
    }
}
struct Tnode *delete(struct Tnode *root)
{
    int ele;
    printf("Enter The Element You want to Delete : ");
    scanf("%d",&ele);
    temp=findtemp(root,ele);
    if(temp!=NULL)
    {
        if(temp->right!=NULL)
        {
            parent=temp;
            temp1=temp->right;
            if(temp1->left==NULL)
            {
                printf("Deleted element is %d ",temp->e);
                temp->e=temp1->e;
                parent->right=temp1->right;
            }
            else{
                do
                {
                    parent=temp1;
                    temp1=temp1->left;
                }while(temp1->left!=NULL);
                printf("Deleted element is %d ",temp->e);
                temp->e=temp1->e;
                parent->left=temp1->right;
            }
        }
        else
        {
            if(temp==root)
                root=temp->left;
            else
            {
                parent=findparent(root,temp);
                if(temp->e<parent->e)
                    parent->left=temp->left;
                else
                    parent->right=temp->left;
            }
        }
    }
    else
        printf("Tree is Empty :-)\n");
    return root;
}
void display(struct Tnode *root)

```

```

{
    if(root==NULL)
        printf("The tree is Empty\n");
    else{
        struct Queue *Q;
        Q=(struct Queue *)malloc(sizeof(struct Queue));
        Q->front=Q->rear=-1;
        Q->max=100;
        enqueue(Q,root);
        while(isempty(Q))
        {
            struct Tnode *temp2=NULL;
            temp2=dequeue(Q);
            printf(" %d ",temp2->e);
            if(temp2->left!=NULL)
                enqueue(Q,temp2->left);
            else{
            }
            if(temp2->right!=NULL)
                enqueue(Q,temp2->right);
            else{
            }
        }
        printf("\n");
    }
}
void inorder(struct Tnode *root)
{
    if(root==NULL){
    }
    else{
        inorder(root->left);
        printf(" %d ",root->e);
        inorder(root->right);
    }
}
void preorder(struct Tnode *root)
{
    if(root==NULL){
    }
    else{
        printf(" %d ",root->e);
        preorder(root->left);
        preorder(root->right);
    }
}
void postorder(struct Tnode *root)
{
    if(root==NULL){
    }
    else{
        postorder(root->left);
        postorder(root->right);
        printf(" %d ",root->e);
    }
}
int main(void)
{
    int op,ele;
    do
    {
        printf("\t\t\t***Choose any Option***\n");
        printf("\t1.Insert  2.find  3.Delete  4.Display  5.Inorder  6.Preorder  7.Postorder\n");
        scanf("%d",&op);
        switch(op)
        {
            case 1 :root=insert(root);
                    break;
            case 2 :find(root);
                    break;
            case 3 :root=delete(root);

```

```
                break;
            case 4:display(root);
                break;
            case 5: inorder(root);
                break;
            case 6: preorder(root);
                break;
            case 7:postorder(root);
                break;
            case 8: break;
        default    : printf("\n***Invalid Choice***\n");
    }
}while(op!=8);
}
```