BINARY SEARCH TREE AND ITS ALL OPERATINS BY – GAURAV RAJPUT

```
#include<iostream>
#include<queue>
using namespace std;

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

//create a node and store value and left,right pointer is NULL

tree* getnode(int data)
{
        tree* newnode = new tree();
   newnode->data = data;
   newnode->left = newnode->right = NULL;
   return newnode;
}

// Insert new node
```

Designed By: Gaurav Rajput

```
tree* insert(tree* root,int data)
{
        if(root==NULL)
        {
                root = getnode(data);
                return root;
        }
        else if(data == root->data)
        return root;
        else if(data < root->data)
        {
                root->left = insert(root->left,data);

        }
        else
        {
                root->right = insert(root->right,data);

        }
}

// find minmum of value
```

Designed By: Gaurav Rajput

```
int findmin(tree* root)
{
        if(root==NULL)
        return -1;
        while(root->left!=NULL)
        {
                root = root->left;
        }
        return root->data;
}

// find address of minmum value

tree* Findmin(tree* root)
{
        if(root==NULL)
        return NULL;
        while(root->left!=NULL)
        {
                root = root->left;
        }
        return root;
```

Designed By: Gaurav Rajput

```
}

// Delete node

tree* deletenode(tree* root,int data)
{
        if(root==NULL)
        return root;
        else if(data < root->data)
        root->left = deletenode(root->left,data);
        else if(data > root->data)
        root->right = deletenode(root->right,data);
        else
        {
                if(root->left == NULL && root->right == NULL)
                {
                        delete root;
                        root = NULL;
                }
                else if(root->left == NULL)
                {
                        tree* temp = root;
                        root = root->right;
```

Designed By: Gaurav Rajput

```
                delete temp;
            }
            else if(root->right == NULL)
            {
                tree* temp = root;
                root = root->left;
                delete temp;
            }
            else
            {
                tree* temp = Findmin(root->right);
                root->data =  temp->data;
                root->right = deletenode(root->right,temp->data);
            }
        }
        return root;
}

// search a node

bool search(tree* root,int data)
{
        if(root==NULL) return false;
```

Designed By: Gaurav Rajput

```
   else if(root->data==data)
        return true;
        else if(data <=root->data)
        return search(root->left,data);
        else
        search(root->right,data);
}

//preorder travesal

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

//postorder traversal

void postorder(tree* root)
{
        if(root==NULL) return;
```

Designed By: Gaurav Rajput

```
        postorder(root->left);

        postorder(root->right);
        printf(" %d ->",root->data);
}

// Inorder traversal

void inorder(tree* root)
{
        if(root==NULL) return;

        inorder(root->left);
        printf(" %d ->",root->data);
        inorder(root->right);
}

// count number of node

int countnode(tree* root, int count)
{
        if(root==NULL)
```

Designed By: Gaurav Rajput

```
        return count;
        else
        {
        return 1+countnode(root->left,count) + countnode(root->right,count);
        }
}

// Level order travesal

void levelorder(tree* root)
{
        if(root==NULL) return;
        queue<tree*> Q;
        Q.push(root);
        while(!Q.empty())
        {
                tree* current = Q.front();
                printf("%d -> ",current->data);
                if(current->left!=NULL) Q.push(current->left);
                if(current->right!=NULL) Q.push(current->right);
                Q.pop();
        }
}
```

Designed By: Gaurav Rajput

```
// Find maxmum value in tree

int findmax(tree* root)
{
        if(root==NULL)
        return -1;
        while(root->right!=NULL)
        {
                root = root->right;
        }
        return root->data;
}

// Find hight of tree

int  hightoftree(tree* root)
{
        if(root==NULL)
        return -1;

        return 1+max(hightoftree(root->left),hightoftree(root->right));
}
```

Designed By: Gaurav Rajput

```
// Main function


int main()

{

        tree* root = NULL;

        cout<<"*****************************\tDESIGNED BY : GAURAV
RAJPUT\t*********************************"<<endl;

        cout<<"                        -------------------------                "<<endl;

         int count = 0;

        int n=1,val;

        while(n!=12)

        {

                        cout<<"\t\t              ~:MENU:~ \n\t\t 1. Insert \n\t\t 2.
Search \n\t\t 3. Preorder traversal\n\t\t ";

                        cout<<"4. Post order Traversal\n\t\t 5. Inorder
traversal\n\t\t ";

                        cout<<"6. Count number of nodes\n\t\t 7. Levelorder
traversal\n\t\t 8. Hight of tree\n\t\t";

                        cout<<" 9. Delete node\n\t\t 10. Minmum of tree\n\t\t 11.
Maximum of tree\n\t\t 12. Exit \n\t\t ";

                        cin>>n;

                switch(n){

                        case 1:cout<<"\t\t Enter node value if node are insert then
enter -1 for exit"<<endl;

                        cout<<"\t\t ";
```

Designed By: Gaurav Rajput

```
            cin>>val;
            while(val!=-1){



            root = insert(root,val);
            cout<<"\t\t ";
            cin>>val;
        }
            break;
            case 2: cout<<"\t\t Enter search value : ";
            cin>>val;
            if(search(root,val)==true) cout<<"\t\t Found"<<endl;
else
cout<<"\t\t Not found"<<endl;
break;
case 3: cout<<"\t\t Preorder traversal:\n\t\t ";
preorder(root);
cout<<endl;
break;
case 4: cout<<"\t\t Postorder traversal:\n\t\t ";
postorder(root);
cout<<endl;
break;
```

Designed By: Gaurav Rajput

```
            case 5: cout<<"\t\t Inorder traversal:\n\t\t";
            inorder(root);
            cout<<endl;
            break;
            case 6: count =0;
            count = countnode(root,count);
            cout<<"\t\t Number of nodes = "<<count<<endl;
            break;
            case 7:
                        cout<<"\t\t ";
                         levelorder(root);
            cout<<"\n\t\t ";
            break;
            case 8: cout<<"\t\t Hight of tree is = "<<hightoftree(root)<<endl;
            break;
            case 9:cout<<"\t\t Enter delete element = ";
            cin>>count;
            root = deletenode(root,count);
            break;
            case 10: cout<<"\t\t Minmum value of tree is =
"<<findmin(root)<<endl;
            break;
            case 11: cout<<"\t\t Maximum value of tree is =
"<<findmax(root)<<endl;
```

Designed By: Gaurav Rajput

```
            break;
            default: exit(0);
               }
        }

}
```

Designed By: Gaurav Rajput