Question 1

a)

#include<iostream>

using namespace std;

int arr[100]={0};

void inorder(int pos)

{

int i=(2\*pos)+1,j=(2\*pos)+2;

if(arr[i]!=0)inorder(i);

cout<<arr[pos]<<" ";

if(arr[j]!=0)inorder(j);

}

void preorder(int pos)

{

int i=2\*pos+1,j=2\*pos+2;

cout<<arr[pos]<<" ";

if(arr[i]!=0)preorder(i);

if(arr[j]!=0)preorder(j);

}

void postorder(int pos)

{

int i=2\*pos+1,j=2\*pos+2;

if(arr[i]!=0)postorder(i);

if(arr[j]!=0)postorder(j);

cout<<arr[pos]<<" ";

}

int main()

{

int count=0;

int d,i;

int c=1;

while(c)

{

cout<<"enter 1 to insert element\nenter 2 to delete\nenter 3 to search\nenter 4 for preorder traversal\nenter 5 for inorder traversal\nenter 6 for postorder traversal\nenter your choice ";

cin>>c;

switch(c)

{

case 1:

cout<<"\nenter data ";

cin>>d;

arr[count++]=d;

break;

case 2:

cout<<"\nenter the element to delete ";

cin>>d;

for( i=0;i<count;i++)

{

if(arr[i]==d)

{

arr[i]=0;

break;

}

}

if(i==count)cout<<"\nthe element is not found \n";

break;

case 3:

cout<<"\nenter the element to search ";

cin>>d;

for( i=0;i<count;i++)

{

if(arr[i]==d)

{

cout<<"\npresent : \n";

break;

}

}

if(i==count)cout<<"the element is not found \n ";

break;

case 4:

cout<<"preorder : ";

preorder(0);

cout<<"\n";

break;

case 5:

cout<<"inorder : ";

inorder(0);

cout<<"\n";

break;

case 6:

cout<<"postorder : ";

postorder(0);

cout<<"\n";

break;

}

}

}

b)

#include<iostream>

using namespace std;

template<typename t>

class tree

{

public:

t data;

tree \* right;

tree \* left;

tree(t d)

{

this->data=d;

this->left=NULL;

this->right=NULL;

}

tree \* create (tree \*& root)

{

t d ;

cout<<"enter the data : ";

cin>>d;

root=new tree (d);

if(d==-1)

{

return NULL;

}

cout<<"for the left of "<<d<<" : ";

root->left=create(root->left);

cout<<"for the right of "<<d<<" : ";

root->right=create(root->right);

return root;

}

void inorder( tree \* root)

{

if(root==NULL)return ;

inorder(root->left);

cout<<root->data<<" ";

inorder(root->right);

}

};

int main()

{

tree <int> \* root;

root=root->create(root);

cout<<"\n inorder : ";

root->inorder(root);

}

Q2

#include<iostream>

using namespace std;

template<typename t>

class tree

{

public:

t data;

tree \* right;

tree \* left;

tree(t d)

{

this->data=d;

this->left=NULL;

this->right=NULL;

}

tree \* create (tree \*& root)

{

t d ;

cout<<"enter the data : ";

cin>>d;

root=new tree (d);

if(d==-1)

{

return NULL;

}

cout<<"for the left of "<<d<<" : ";

root->left=create(root->left);

cout<<"for the right of "<<d<<" : ";

root->right=create(root->right);

return root;

}

void height(tree \* root, int h, int & ans)

{

if(root==NULL) {ans=max(ans,h); return ;}

height(root->left,h+1,ans);

height(root->right,h+1,ans);

}

void inorder( tree \* root)

{

if(root==NULL)return ;

inorder(root->left);

cout<<root->data<<" ";

inorder(root->right);

}

};

int main()

{

tree <int> \* root;

root=root->create(root);

cout<<"\n inorder : ";

root->inorder(root);

int ans=-25;

root->height(root,0,ans);

cout<<"height : "<<ans;

}

Q3

#include<iostream>

#include<queue>

using namespace std;

template<typename t>

class tree

{

public:

t data;

tree \* left;

tree \* right;

tree (t d)

{

this->data=d;

this->left=NULL;

this->right=NULL;

}

tree<t> \* create(tree<t> \* root)

{

t d;

cout<<"enter the data (-1 for NULL): ";

cin>>d;

if(d==-1)

{

return NULL;

}

root=new tree(d);

cout<<"for the left of "<<d<<" ";

root->left=create(root->left);

cout<<"for the right of "<<d<<" ";

root->right=create(root->right);

return root;

}

void deletion(tree<t> \* & root, tree <t>\* p)

{

queue<tree\*> q;

q.push(root);

while(!q.empty())

{

tree<t> \* temp=q.front();

q.pop();

if(temp==p)

{

root=NULL;

return;

}

if(temp->right)

{

if(temp->right==p)

{

temp->right=NULL;

return;

}

else

{

q.push(temp->left);

}

}

if(temp->left)

{

if(temp->left==p)

{

temp->left=NULL;

return;

}

else

{

q.push(temp->left);

}

}

}

}

void del(tree<t> \* & root, int key)

{

queue<tree\*> q;

q.push(root);

tree<t> \* key\_node=NULL;

tree <t>\* temp=NULL;

while(!q.empty())

{

temp=q.front();

q.pop();

if(temp->data==key)

{

key\_node=temp;

}

if(temp->left) q.push(temp->left);

if(temp->right) q.push(temp->right);

}

if(key\_node!=NULL)

{

int x=temp->data;

deletion(root,temp);

key\_node->data=x;

}

}

void inorder( tree<t>\* &root)

{

if(root==NULL)return ;

inorder(root->left);

cout<<root->data<<" ";

inorder(root->right);

}

};

int main()

{

tree<int> \* root;

root=root->create(root);

root->inorder(root);

int n;

cout<<"\nenter the data to delete : ";

cin>>n;

root->del(root,n);

cout<<endl<<"after deletion inorder : ";

root->inorder(root);

}

Question 4

#include<iostream>

using namespace std;

template<typename t>

class tree

{

public:

t data;

tree \* right;

tree \* left;

tree(t d)

{

this->data=d;

this->left=NULL;

this->right=NULL;

}

tree \* create (tree \*& root)

{

t d ;

cout<<"enter the data : ";

cin>>d;

root=new tree (d);

if(d==-1)

{

return NULL;

}

cout<<"for the left of "<<d<<" : ";

root->left=create(root->left);

cout<<"for the right of "<<d<<" : ";

root->right=create(root->right);

return root;

}

void height(tree \* root, int h, int & ans)

{

if(root==NULL) {ans=max(ans,h); return ;}

height(root->left,h+1,ans);

height(root->right,h+1,ans);

}

void inorder( tree \* root)

{

if(root==NULL)return ;

inorder(root->left);

cout<<root->data<<" ";

inorder(root->right);

}

};

int main()

{

tree <int> \* root;

root=root->create(root);

cout<<"\n inorder : ";

root->inorder(root);

int ans=-25;

root->height(root,0,ans);

cout<<"height : "<<ans;

}

Q5

#include<iostream>

using namespace std;

template<typename t>

class tree

{

public:

t data;

tree \* right;

tree \* left;

tree(t d)

{

this->data=d;

this->left=NULL;

this->right=NULL;

}

tree \* create (tree \*& root)

{

t d ;

cout<<"enter the data : ";

cin>>d;

root=new tree (d);

if(d==-1)

{

return NULL;

}

cout<<"for the left of "<<d<<" : ";

root->left=create(root->left);

cout<<"for the right of "<<d<<" : ";

root->right=create(root->right);

return root;

}

void depth(tree \* root, int h, int & ans,int key)

{

if(root==NULL) { return ;}

height(root->left,h+1,ans,key);

if(root->data==key) {ans=h; return;}

height(root->right,h+1,ans,key);

}

void inorder( tree \* root)

{

if(root==NULL)return ;

inorder(root->left);

cout<<root->data<<" ";

inorder(root->right);

}

};

int main()

{

tree <int> \* root;

root=root->create(root);

cout<<"\n inorder : ";

root->inorder(root);

int ans=-1;

int f;

cout<<"enter the data whose depth is to find : ";

cin>>f;

root->depth(root,0,ans,f);

cout<<"depth : "<<ans;

}

Q6

#include<iostream>

using namespace std;

class tree

{

public:

int data;

tree \* left;

tree \* right;

tree (int d)

{

this->data=d;

this->left=NULL;

this->right=NULL;

}

void inorder(tree \*& root)

{

if(root==NULL)

{

return ;

}

if(root->left)inorder(root->left);

cout<<root->data<<" ";

if(root->right) inorder(root->right);

}

};

int search(int in[],int s,int e, int key)

{

for(int i=s;i<=e;i++)

{

if(in[i]==key) return i;

}

return -1;

}

int \* c(int in[],int s,int e,int n,int pre[])

{

int \* ans=new int [e-s+1];

int k=0;

for(int i=0;i<n;i++)

{

if(search(in,s,e,pre[i])!=-1)

{

ans[k++]=pre[i];

}

}

return ans;

}

tree \* construct ( int in[] , int pre [],int s,int e,int n )

{

if(s>e)

{

return NULL;

}

tree \* root =new tree(pre[0]);

if(s==e)

{

return root;

}

int index=search(in,s,e,pre[0]);

int \* lp=c(in,s,index-1,n,pre);

int \* rp=c(in,index+1,e,n,pre);

root->left=construct(in,lp,s,index-1,index-s);

root->right=construct(in,rp,index+1,e,e-index);

return root;

}

int main()

{

int n;

cout<<"enter the value of n : ";

cin>>n;

int in[n];

int pre[n];

cout<<"inorder : ";

for(int i=0;i<n;i++)

{

cin>>in[i];

}

cout<<"preorder : ";

for(int i=0;i<n;i++)

{

cin>>pre[i];

}

tree \* root=construct(in,pre,0,n-1,n);

root->inorder(root);

}