BST INSERTION:

#include<iostream>

using namespace std;

struct node\* newnode(int);

struct node\* insert(struct node\*, int);

struct node

{

int data;

struct node\* left;

struct node\* right;

};

struct node\* newnode(int data)

{

struct node\* new\_node=new node();

new\_node->data=data;

new\_node->left=NULL;

new\_node->right=NULL;

return new\_node;

}

struct node\* insert(struct node\* root, int n)

{

if(root==NULL)

{

return newnode(n);

}

if(n<root->data)

{root->left=insert(root->left,n);}

else if(n>root->data)

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

return root;

}

int main()

{

struct node\* root=NULL;

root=insert(root,1);

insert(root,30);

insert(root,20);

insert(root,40);

insert(root,70);

insert(root,60);

insert(root,80);

return 0;

}

Ques-1:

[Height of Binary Tree](https://practice.geeksforgeeks.org/problems/height-of-binary-tree/1)

Height of Binary Tree:

int height(struct node\* root)

{

if(root!=NULL){

lefth=height(root->left);

righth=height(root->right);

if(lefth>righth)

return (lefth+1);

else

return (righth+1);

}

else

return 0;

}

Ques-2:

Inorder,preorder,postorder

Inorder

void inorder(struct node\* root)

{

if(root!=NULL)

{

inorder(root->left);

cout<<root->data;

inorder(root->right);}}

Preorder

void preorder(struct node\* root)

{

if(root!=NULL)

{

cout<<root->data;

preorder(root->left);

preorder(root->right);

}}

Postorder

void postorder(struct node\* root)

{

if(root!=NULL){

postorder(root->left);

postorder(root->right);

cout<<root->data;

}

}

Ques-3

(Search a key in BST):

bool searchkey(struct node\* root,int data)

{

if(root!=NULL){

if (data==root->data)

return true;

if(data<root->data)

{

searchkey(root->left,data);

}else

if(data>root->data)

{

searchkey(root->right,data);

}

}else

return false;

}

Ques-4(Height of a Binary tree)

int lefth=0,righth=0;

int height(struct node\* root)

{

if(root==NULL)

return 0;

lefth=height(root->left);

righth=height(root->right);

if(lefth>righth)

return (lefth+1);

Ques-5(Level order Traversal)

#include<iostream>

using namespace std;

struct node\* newnode(int);

struct node\* insert(struct node\*, int);

int height(struct node\*);

void nextLevel(struct node\*);

void printLevel(struct node\*,int);

struct node

{

int data;

struct node\* left;

struct node\* right;

};

int lefth=0,righth=0;

struct node\* newnode(int data)

{

struct node\* new\_node=new node();

new\_node->data=data;

new\_node->left=NULL;

new\_node->right=NULL;

return new\_node;

}

struct node\* insert(struct node\* root, int n)

{

if(root==NULL)

{

return newnode(n);

}

if(n<root->data)

{

root->left=insert(root->left,n);

}

else if(n>root->data)

{

root->right=insert(root->right,n);

}

return root;

}

void nextLevel(struct node\* root)

{

int h,i;

h=height(root);

for(i=1;i<=h;i++)

printLevel(root,i);

}

void printLevel(struct node\* root,int level)

{

if(root==NULL)

return ;

if(level==1)

cout<<root->data;

else if(level>1)

{

printLevel(root->left,level-1);

printLevel(root->right,level-1);

}

}

int height(struct node\* root)

{

if(root==NULL)

return 0;

lefth=height(root->left);

righth=height(root->right);

if(lefth>righth)

return (lefth+1);

else return (righth+1);

}

int main()

{

struct node\* root=NULL;

int key;

root=insert(root,1);

insert(root,30);

insert(root,20);

insert(root,40);

insert(root,70);

insert(root,60);

insert(root,80);

nextLevel(root);

return 0;

}

Ques -6:(Left View of TREE)

Solution-

Approach:print only one node and first node in only one level

For Left View all things will be same as previous Question

(Level order Traversal) except for one function Mentioned

Below:-

int printGivenLevel(node\* root, int level)

{ int x =0;

if (root == NULL)

return 0;

if (level == 1)

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

return 1;}

else if (level > 1)

{ if(x!=1)

x=printGivenLevel(root->left, level-1);

if(x!=1)

x=printGivenLevel(root->right, level-1);

return x;

}

}

Ques -7:(**Count Leaves in Binary Tree** )

[Number of leaf nodes](https://practice.geeksforgeeks.org/problems/count-leaves-in-binary-tree/1)

int count=0;//public

void countleaves(struct node\* root)

{

if(root!=NULL)

{

inorder(root->left);

if(root->left==NULL && root->right==NULL)

count++;

inorder(root->right);

}}

Ques -8:(Vertical Order Traversal)

[Print a Binary Tree in Vertical Order](https://practice.geeksforgeeks.org/problems/print-a-binary-tree-in-vertical-order/1)

#include<iostream>

using namespace std;

struct node\* createnode(int);

struct node\* insert(struct node\*,int);

void getminmax(struct node\*, int\* , int\*, int);

void printvertical(struct node\* ,int,int);

void getVertical(struct node\*);

struct node

{

int data;

node\* left;

node\* right;

};

struct node\* createnode(int data)

{

struct node\* new\_node=new node();

new\_node->data=data;

new\_node->left=NULL;

new\_node->right=NULL;

return new\_node;

}

struct node\* insert(struct node\* root, int data)

{

if(root==NULL)

{

return createnode(data);

}

if(data<root->data)

{

root->left=insert(root->left,data);

}

else if(data>root->data)

{

root->right=insert(root->right,data);

}

return root;

}

void inorder(struct node\* root)

{

if(root!=NULL){

inorder(root->left);

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

inorder(root->right);

}

}

void getVertical(struct node\* root)

{

int i;

int min=0,max=0;

getminmax(root,&min,&max,0);

for(i=min;i<=max;i++)

{

printvertical(root,i,0);

}

}

void getminmax(struct node\* root,int \*leftmin,int \*rightmax,int dist)

{

if(root!=NULL) {

if(\*leftmin>dist)

{

\*leftmin=dist;

}

if(\*rightmax<dist)

{

\*rightmax=dist;

}

getminmax(root->left,leftmin,rightmax,dist-1);

getminmax(root->right,leftmin,rightmax,dist+1);

}

}

void printvertical(struct node\* root,int width,int dist)

{

if(root!=NULL){

if(width==dist)

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

printvertical(root->left,width,dist-1);

printvertical(root->right,width,dist+1);

}

}

int main()

{

int a=0,b=0;

struct node\* root=NULL;

root=insert(root,10);

insert(root,9);

insert(root,15);

insert(root,6);

insert(root,7);

insert(root,25);

insert(root,14);

//inorder(root);

getVertical(root);

//getminmax(root,&a,&b,0);

//

return 0;

}

Ques -9:(Top View of Binary Tree)

Approach:print only one node and top node in only one column

For Top View all things will be same as previous Question

(Vertical Order Traversal) except for one function Mentioned

Below:-

int printvertical(struct node\* root,int width,int dist)

{

int x =0;

if(root!=NULL){

if(width==dist)

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

return 1;}

if(x!=1)

printvertical(root->left,width,dist-1);

if(x!=1)

printvertical(root->right,width,dist+1);

return x;

}

}