# Program Height of Binary Tree

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

struct node

      {
      int key;
      struct node*left,*right;
      };

      struct node *newNode(int item)
      {
      struct node*temp=(struct node*)malloc(sizeof(struct node));
      temp->key=item;
      temp->left=temp->right=NULL;
      return temp;
      }

      void inorder(struct node*root)
      {
      if(root!=NULL)
      {
      inorder(root->left);
      cout<<root->key<<"->"<<endl;
      inorder(root->right);
      }
      }

      void preorder(struct node*root)
      {
      if(root!=NULL)
      {
      cout<<root->key<<"->"<<endl;
      preorder(root->left);
      preorder(root->right);
      }
      }

      void postorder(struct node*root)
      {
      if(root!=NULL)
      {
      postorder(root->left);
      postorder(root->right);
      cout<<root->key<<"->"<<endl;
      }
      }

      struct node*insert(struct node*node,int key)
      {
      if(node==NULL)return newNode(key);
      if(key<node->key)
            node->left=insert(node->left,key);
      else
            node->right=insert(node->right,key);
            return node;
      }

      int maxDepth(node*node)
      {
            if(node==NULL)
```

```
                return 0;
            else
            {
            int lDepth=maxDepth(node->left);
            int rDepth=maxDepth(node->right);

            if (lDepth>rDepth)
                  return(lDepth +1);
            else
                  return(rDepth +1);
            }
      }


      int main()
      {
      struct node*root=NULL;
      root=insert(root,50);
      root=insert(root,30);
      root=insert(root,20);
      root=insert(root,40);
      root=insert(root,70);
      root=insert(root,60);
      root=insert(root,80);

      cout<<"lnorder traversal: ";
      inorder(root);

      cout<<"preorder traversal: ";
      preorder(root);

      cout<<"postorder traversal: ";
      postorder(root);



      cout<<"\n Height of tree is"<< maxDepth(root);
      }
```

```
Output:
lnorder traversal: 20->
30->
40->
50->
60->
70->
80->
preorder traversal: 50->
30->
20->
40->
70->
60->
80->
postorder traversal: 20->
40->
30->
60->
80->
70->
50->

 Height of tree is3
```