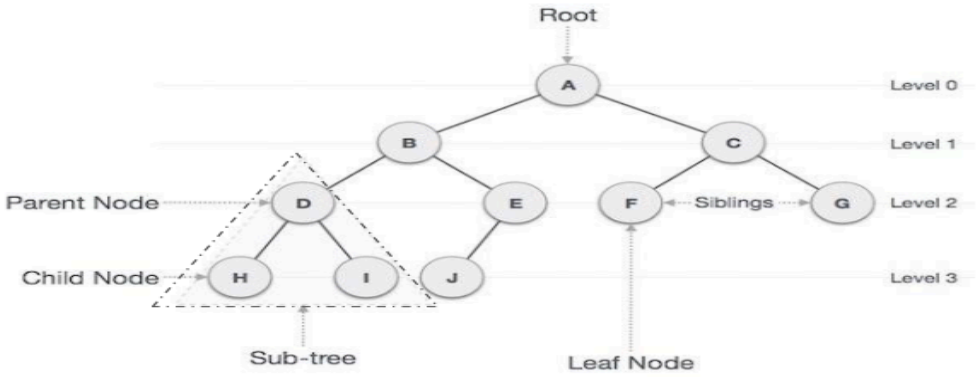
Q.No	Question
1	What is Binary Tree? Explain representation and properties of a binary tree?

Terminology	Description	Example From Diagram
Root	Root is a special node in a tree. The entire tree originates from it. It does not have a parent.	Node A
Parent Node	Parent node is an immediate predecessor of a node.	B is parent of D & E
Child Node	All immediate successors of a node are its children.	D & E are children of B
Leaf	Node which does not have any child is called as leaf	H, I, J, F and G are leaf nodes
Edge	Edge is a connection between one node to another. It is a line between two nodes or a node and a leaf.	Line between A & B is edge
Siblings	Nodes with the same parent are called Siblings.	D & E are siblings
Path / Traversing	Path is a number of successive edges from source node to destination node.	A – B – E – J is path from node A to E
Height of Node	Height of a node represents the number of edges on the longest path between that node and a leaf.	A, B, C, D & E can have height. Height of A is no. of edges between A and H, as that is the longest path, which is 3. Height of C is 1
Levels of node	Level of a node represents the generation of a node. If the root node is at level 0, then its next child node is at level 1, its grandchild is at level 2, and so on	Level of H, I & J is 3. Level of D, E, F & G is 2
Degree of Node	Degree of a node represents the number of children of a node.	Degree of D is 2 and of E is 1
Sub tree	Descendants of a node represent subtree.	Nodes D, H, I represent one subtree.

Binary tree:in which each node can have atmost 2 childern(0 to 2 childern)



2 Write an algorithm for in-order, pre-order, post-order traversal of a binary tree. Explain with an example

Algorithm inorder:

```
Inorer(struct node *root)//L-P-R
{
If(root==NULL)
{
Printf("no nodes are presents");
}
else
{
inrder(root->left);
printtf("%d-->",root->data);
inorder(root->right);
}
```

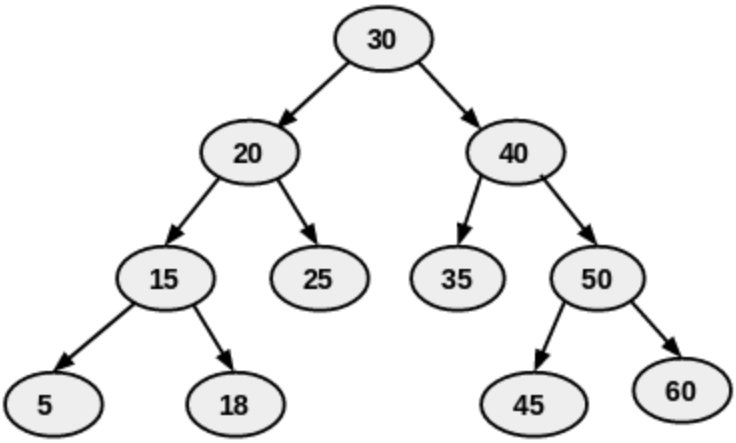
Algorithm preorder:

```
preorer(struct node *root)//P-L-R
{
If(root==NULL)
{
printf("no nodes are presents");
}
else
{
printf("%d--->",root->data);
preorder(root->left);
preorder(root->right);
}
```

Algorithm postorder:

```
postorer(struct node *root)//L-R-P
{
If(root==NULL)
{
printf("no nodes are presents");
}
else
{
postorder(root->left);
postorder(root->right);
printf("%d--->",root>data);
}
```

Example of inorder traversal



- we start recursive call from 30(root) then move to 20 (20 also have sub tree so apply in order on it),15 and 5.
- 5 have no child .so print 5 then move to it's parent node which is 15 print and then move to 15's right node which is 18.
- 18 have no child print 18 and move to 20 .print 20 then move it right node which is 25 .25 have no subtree so print 25.
- print root node 30 .
- now recursively traverse to right subtree of root node . so move to 40. 40 have subtree so traverse to left subtree of 40.
- left subtree of 40 have only one node which is 35. 35 had no further subtree so print 35. move to 40 and print 40.
- traverse to right subtree of 40. so move to 50 now have subtree so traverse to left subtree of 50 .move to 45 , 45 have no further subtree so print 45.
- move to 50 and print 50. now traverse to right subtree of 50 hence move to 60 and print 60.

our final output is {5 , 15 , 18 , 20 , 25 , 30 , 35 , 40 , 45 , 50 , 60}

Example of preorder traversal

- Start with root node 30 .print 30 and recursively traverse the left subtree.
- next node is 20. now 20 have subtree so print 20 and traverse to left subtree of 20 .
- next node is 15 and 15 have subtree so print 15 and traverse to left subtree of 15.
- 5 is next node and 5 have no subtree so print 5 and traverse to right subtree of 15.
- next node is 18 and 18 have no child so print 18 and traverse to right subtree of 20.
- 25 is right subtree of 20 .25 have no child so print 25 and start traverse to right subtree of 30.

- next node is 40. node 40 have subtree so print 40 and then traverse to left subtree of 40.
- next node is 35. 35 have no subtree so print 35 and then traverse to right subtree of 40.
- next node is 50. 50 have subtree so print 50 and traverse to left subtree of 50.
- next node is 45. 45 have no subtree so print 45 and then print 60(right subtree) of 50.

our final output is {30 , 20 , 15 , 5 , 18 , 25 , 40 , 35 , 50 , 45 , 60}

Example of postorder traversal

- We start from 30, and following Post-order traversal, we first visit the left subtree 20. 20 is also traversed post-order.
- 15 is left subtree of 20 .15 is also traversed post order.
- 5 is left subtree of 15. 5 have no subtree so print 5 and traverse to right subtree of 15 .
- 18 is right subtree of 15. 18 have no subtree so print 18 and then print 15. post order traversal for 15 is finished.
- next move to right subtree of 20.
- 25 is right subtree of 20. 25 have no subtree so print 25 and then print 20. post order traversal for 20 is finished.
- next visit the right subtree of 30 which is 40 .40 is also traversed post-order(40 have subtree).
- 35 is left subtree of 40. 35 have no more subtree so print 35 and traverse to right subtree of 40.
- 50 is right subtree of 40. 50 should also traversed post order.
- 45 is left subtree of 50. 45 have no more subtree so print 45 and then print 60 which is right subtree of 50.
- next print 50 . post order traversal for 50 is finished.
- now print 40 ,and post order traversal for 40 is finished.
- print 30. post order traversal for 30 is finished.

our final output is {5 , 18 , 15 , 25 , 20 , 35 , 45 , 60 , 50 , 40 , 30}

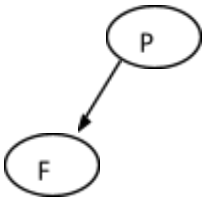
3

Define binary search tree. Construct the binary search Tree for the below given data.

P, F , B, H, G , S, R, Y, T, W, Z

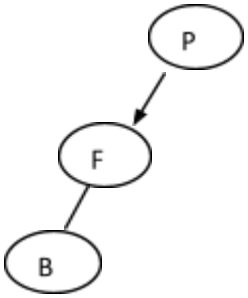
STEP1:THE FIRST ELEMENT P IS INSERTED AS A ROOT NODE

Step2:to insert F
We have compare with root node:P
F<P ----T or F>P
Then we have to verify p->left is NULL or not---Y
We have to place F at left side P

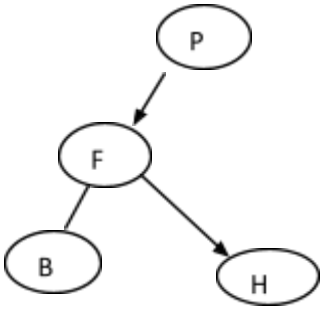


Step3: to insert B

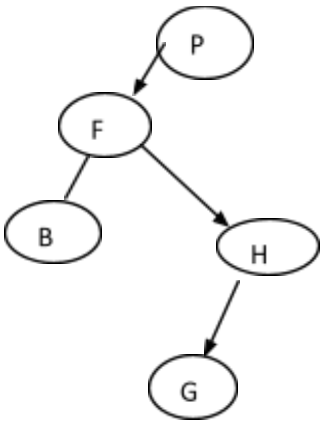
We have compare with root node---P
B<P ----T B>P then we have check p->left==NULL----no
Then we will move to P->left(F)
B<F ---T B>F then we have to verify F->left=NULL----yes
We have to place B at the left side of F F->left=B



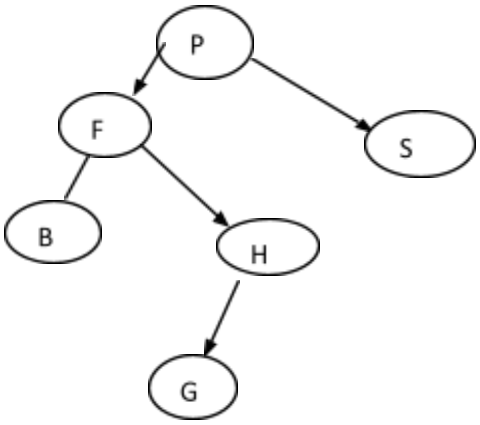
Step4:to insert H in to BST
H<P ---T H>P p->left==NULL---no
Then we have to move to p->left(F)
H<F H>F----T F->right==NULL---yes F->right=H



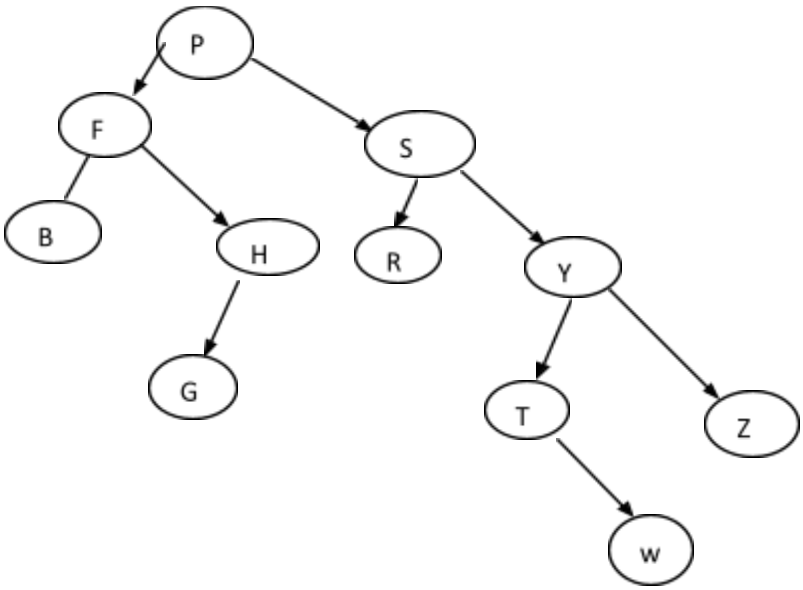
Step5:to insert G compare with root(P)
G<P----T G>P we will move left side of P
P->left==NULL---no we will move p->left(F)
Now we have start comparing with F
G<F G>F---T then we have to verify F->right==NULL--no
Then we will mve to F->right(H)
We start comparing with H
G<H----T G>H then we will move to left side of H H->left==NULL--yes
H->left=G



Step6: to insert S
We compare with root(P)
S<P S>P---T then we have to verify P->right==NULL--yes
P->right=S



Step7:to insert R
R<P R>P----T then we have to verify P->right==NULL---no
We will move to P->right(S)
R<S ---T R>S then we have to verify S->left==NULL---yes
S->left=R



4 WRITE A C PROGRAM TO IMPLEMENT OPERATIONS OF A BINARY SEARCH TREE

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *left,*right;
}*new,*temp,*root,*prev,*current;
void insert(struct node *root ,struct node *new);
void inorder(struct node *root);
void preorder(struct node *root);
void postorder(struct node *root);
void serch(struct node *root,int key);
void main()
{
    int ele,opt,key;
    root=NULL;
```

```
    do
     {
       printf("1.insert  2.inorder   3.preorder  4.postorder  5.search\n");
       printf("enter u r option");
       scanf("%d",&opt);
       switch(opt)
          {
            case 1:

              printf("\n enter value:");
              scanf("%d",&ele);
                new=(struct node *)malloc(sizeof(struct node));
                new->data=ele;
              new->left=NULL;
              new->right=NULL;
              if(root==NULL)
                   root=new;
              else
                    insert(root,new);
              break;
              case 2:
                  inorder(root);break;
              case 3:
                 preorder(root); break;
              case 4:
                   postorder(root); break;
              case 5:
                   printf("enter the element to search");
                   scanf("%d",&key);
                   printf("%d",search(root,key));
              break;
              default:
                   printf("wrong opt") ;  exit(1);
          }
    }while(1);
}//end of main


void insert(struct node *root,struct node *new)
{

if(new->data < root->data)
{
    if(root->left == NULL)
        root->left=new;
    else
        insert(root->left,new);
}


else if(new->data > root->data)
{
    if( root->right == NULL )
        root->right=new;
```

```
    else
        insert(root->right,new);
}

}//insert


void inorder( struct node * temp)
{
 if(temp != NULL)
 {
    inorder(temp->left);
    printf("%d->",temp->data);
    inorder(temp->right);
 }

}//inorder


void preorder( struct node * temp)
{
 if(temp != NULL)
 {
    printf("%d->",temp->data);
    preorder(temp->left);
    preorder(temp->right);
 }

}//inorder


void postorder( struct node * temp)
{
 if(temp != NULL)
 {
    postorder(temp->left);
    postorder(temp->right);
    printf("%d->",temp->data);
 }

}//inorder


int search(struct node * root, int key)
{
    struct node * temp;
    int flag=0;
    current=root;

    while(current != NULL)
    {

        if( current->data == key)//key-----element to search
        {
            printf("\n the %d element is found ",key);
```

```
            flag=1;
            break;
        }
        else
        {
            prev=current;
            if( key <current ->data)
                current=current->left;
            else
                current=current->right;
        }
    }//while

    if(flag==0)
        printf("\n the element %d is not found",key);
    return flag;
}//search
```

5

Create the binary search tree using the following data elements.

43, 10, 79, 90, 12, 54, 11, 9, 50

WRITE ALL THE STEPS TO INSERT AS ABOVE EAMPLE