NAME-YASH GANDHI SE IT BATCH A 14

EXP 5

QUESTION

1- BINARY TREE

a) Perform the following operations of Binary tree

-Creation of Binary Tree using recursion

-Counting No.of Nodes in a Binary Tree

-Counting No.of leaves in a Binary Tree

-Finding height of a Binary tree

b) Traverse the Binary tree using all traversing techniques

CODE-

```
#include<stdio.h>
#include<stdlib.h>

typedef struct node     //structure declaration
{
 int data;
 struct node* left;
 struct node* right;
}node;

node* create()        //function to create tree
```

```
{
 int m,x;
 node* temp=(node*)malloc(sizeof(node*));
 temp->left=NULL;
 temp->right=NULL;
 printf("Enter data\n");
 scanf("%d",&x);

 if(x==-1)            //No child condition
 {
  return NULL;
 }

  else
  {
    temp->data=x;

  printf("  For left child of %d\n",x);
   temp->left=create();

  printf("  For right child of %d\n",x);
  temp->right=create();
  }

 return temp;
}
```

```
void inorder(node *root)       //function to traverse in inorder l,ro,ri
{
if(root==NULL)
return;
else
{
inorder(root->left);
printf("%d\t",root->data);
inorder(root->right);
}
}

void postorder(node *root)   //postorder l,ri,ro
{
if(root==NULL)
return;
else
{
postorder(root->left);
postorder(root->right);
printf("%d\t",root->data);
}
}

void preorder(node *root)     //preorder ro,l,ri
```

```
{
if(root==NULL)
return;
else
{
printf("%d\t",root->data);
preorder(root->left);
preorder(root->right);
}
}

int leaf(node* root,int i)
{
 if(root->left==NULL&&root->right==NULL)
  return i+1;

 if(root->left!=NULL)
 i=leaf(root->left,i);

  if(root->right!=NULL)
 i=leaf(root->right,i);

 return i;
}

int nodes(node* root,int count)   //function to count total nodes
```

```
{

 if(root->left!=NULL||root->right!=NULL);
 count++;
 if(root->left!=NULL)
 count=nodes(root->left,count);

 if(root->right!=NULL)
 count=nodes(root->right,count);
 return count;
}

int height(node *root)//functon for height of tree
 {
          if(root==NULL)
                    return 0;
          else
          {
               int lh=height(root->left);
            int rh=height(root->right);
            if(lh>rh)
             return(lh+1);
           else
             return(rh+1);
          }
 }
```

```
void main()
{
  int count=0,i=0,h=0;
printf("Let's Create A Binary Tree\n");
  node* root=create();  //creating tree
  printf("\n\nINORDER\n");
  inorder(root);                              //function calls
  printf("\n\nPOSTORDER\n");
  postorder(root);
  printf("\n\nPREORDER\n");
  preorder(root);

  printf("\n\nHEIGHT OF TREE\n");
   h=height(root);
  printf("%d\n",h);

  printf("\nNODE COUNT\n");
  int c=nodes(root,count);
  printf("%d\n",c);
  printf("\nLEAF COUNT\n");
   i=leaf(root,i);
  printf("%d\n",i);
}

/*
```

OUTPUT-

```
Let's Create A Binary Tree
Enter data
1
  For left child of 1
Enter data
2
  For left child of 2
Enter data
-1
  For right child of 2
Enter data
3
  For left child of 3
Enter data
-1
  For right child of 3
Enter data
-1
  For right child of 1
Enter data
4
  For left child of 4
Enter data
5
```

```
-1
  For right child of 2
Enter data
3
  For left child of 3
Enter data
-1
  For right child of 3
Enter data
-1
  For right child of 1
Enter data
4
  For left child of 4
Enter data
5
  For left child of 5
Enter data
-1
  For right child of 5
Enter data
-1
  For right child of 4
Enter data
-1_
```

```
Enter data
-1
  For right child of 4
Enter data
-1


INORDER
2       3       1       5       4

POSTORDER
3       2       5       4       1

PREORDER
1       2       3       4       5

HEIGHT OF TREE
3

NODE COUNT
5

LEAF COUNT
2
_
```

*/