Lab -10 - Binary Tree :-

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

struct node
 {
int info;
struct node * llink;
struct node * rlink;
 };
typedef struct node * NODE;

NODE getnode()
 {
NODE x;
x = (NODE) malloc (sizeof (struct node));
if (x == NULL)
{
printf ("memory not available");
   exit(0);
   }
return x;
}
NODE insert (int item, NODE root)
{
NODE temp, cur, prev;
 char direction [10];
```

```
int i;
temp = getnode();
temp->info = item;
temp->llink = NULL;
temp->rlink = NULL;
if (root == NULL)
    return temp;
printf("give direction to insert\n");
scanf("%s", direction);
prev = NULL;
cur = root;
for (int i=0; i<strlen(direction) &&
                cur != NULL; i++)
{
prev = cur;
if (direction[i] == 'l')
cur = cur->llink;
else
cur = cur->rlink;
}
if (cur != NULL || i != strlen(direction))
{
printf("insertion not possible\n");
free(temp);
return root;
}
if (cur == NULL)
{
```

```
if (direction[i-1] == 'l')
  prev->llink == temp;
  else
  prev->rlink == temp;
 }
 return root;
  }
Void preorder(NODE root)
{
 if(root != NULL)
  {
 printf("the item is %d\n", root->info);
 preorder (root->llink);
 preorder (root->rlink);
  }
 }
void inorder (NODE root)
 {
 if (root != NULL)
  {
  inorder (root->llink);
 printf ("the item is %d\n", root->info);
  inorder (root->rlink);
  }
 }
void postorder (NODE root)
  {
  if (root != NULL)
  {
```

```
postorder(root->llink);
postorder(root->rlink);
printf("the item is %d\n", root->info);
}
}
void display(NODE root, int i)
{
int j;
if(root != NULL)
{
display(root->rlink, i+1);
for(j=1; j<=i; j++)
printf("    ");
printf("%d\n", root->info);
display(root->llink, i+1);
}
}
void main()
{
NODE root = NULL;
int choice, i, item;
for(;;)
{
printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n
5.display\n");
printf("enter choice\n");
scanf("%d", &choice);
switch(choice)
}
```

```
case 1: printf("enter the item\n");
        scanf("%d", &item);
        root = insert(item, root);
        break;
case 2: if (root == NULL)
        {
        printf("tree is empty");
        }
        else
        {
        printf("given tree is\n");
        display(root, 1);
        printf("the preorder traversal is\n");
        preorder(root);
        }
        break;
case 3: if (root == NULL)
        {
        printf("tree is empty");
        }
        else
        {
        printf("given tree is\n");
        display(root, 1);
        printf("the inorder traversal is\n");
        inorder(root);
        }
        break;
```

```
case 4: if (root == NULL)
        {
        printf ("tree is empty");
        }
      else
      {
      printf ("given tree is");
      display (root, 1);
      printf ("the postorder transversal is \n");
      postorder (root);
    }
    break;
case 5: display (root, 1);
        break;
default : exit (0);
  }
 }
}
```