1BM19CS158

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;
}
void freenode(NODE x)
{
 free(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 (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");
 freenode(temp);
 return(root);
}
if (cur==NULL)
if (direction [i-1]=='l')
 prev->llink=temp;
  else
 prev->rlink=temp;
}
 return(root);
}
```

Date
Page

```
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);
 }
 }
 int main ()
 {
 NODE root =NULL;
 int choice, i, item;
 for(;;)
{
printf ("1. Insert \n 2. Preorder \n3. Inorder \n 4. Postorder \n5.Display\n");
printf (" Enter the 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..\n");
        display (root, 1);
        printf(" The postorder traversal is:\n");
        postorder (root);
      }
    break;
case 5: display (root, 1);
        break;
default: printf("Invalid choice entered.\n");
        exit(0);
    }
 }
  return 0;
}
```