Demonstrating binary Tree ~ Mahantesh Gattimani 1BM19EE031

```
#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\n");
        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 the directions 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);
}
void preorder(NODE root)
{
    if(root != NULL)
    {
        printf("%d\n", root->info);
        preorder(root->llink);
        preorder(root->rlink);
    }
}
void inorder(NODE root)
{
    if(root != NULL)
    {
        inorder(root->llink);
        printf("%d\n", root->info);
        inorder(root->rlink);
    }
}
void postorder(NODE root)
{
    if(root != NULL)
    {
        postorder(root->llink);
        postorder(root->rlink); printf("%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
\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\n");
         }
         else
         {
           printf("given tree is");
           display(root, 1);
           printf("the preorder traversal is\n");
           preorder(root);
         }
         break;
case 3 : if(root==NULL)
         {
           printf("tree is empty\n");
         }
         else
         {
           printf("given tree is");
           display(root, 1);
           printf("the inorder-traversal is\n");
           inorder(root);
         }
         break;
case 4 : if(root==NULL)
         {
           printf("tree is empty\n");
         }
         else
         {
```

```
        printf("given tree is");
        display(root, 1);
        printf("the postorder traversal is \n");
        postorder(root);

    }
    break;
    case 5: display(root, 1);
            break;
    default: exit(0);
    }
  }
}
```

Expected output -

1. insert
2. preorder
3. inorder
4. postorder
5. display

enter the choice

1

enter the item

1

1. insert
2. preorder
3. inorder
4. postorder

5. display
enter the choice
1
enter the item
2
give the direction to insert
l

1. insert
2- preorder
3- inorder
4- postorder
5- display
enter the choice
1
enter the item
3
give the direction to insert
1
1. insert
2. preorder
3. inorder
4. postorder
5. display
enter the choice
2
given tree is
3
1
2
the preorder traversal is

1
2
3

1. insert
2. preorder
3. inorder
4. postorder
5. display
enter the choice

3
given tree is

3
1
2
the inorder traversal is
2
1
3

1. insert
2. preorder
3. inorder
4. postorder
5. display
enter the choice

4
given tree is

3
1
2
the post order traversal is
2
3
1