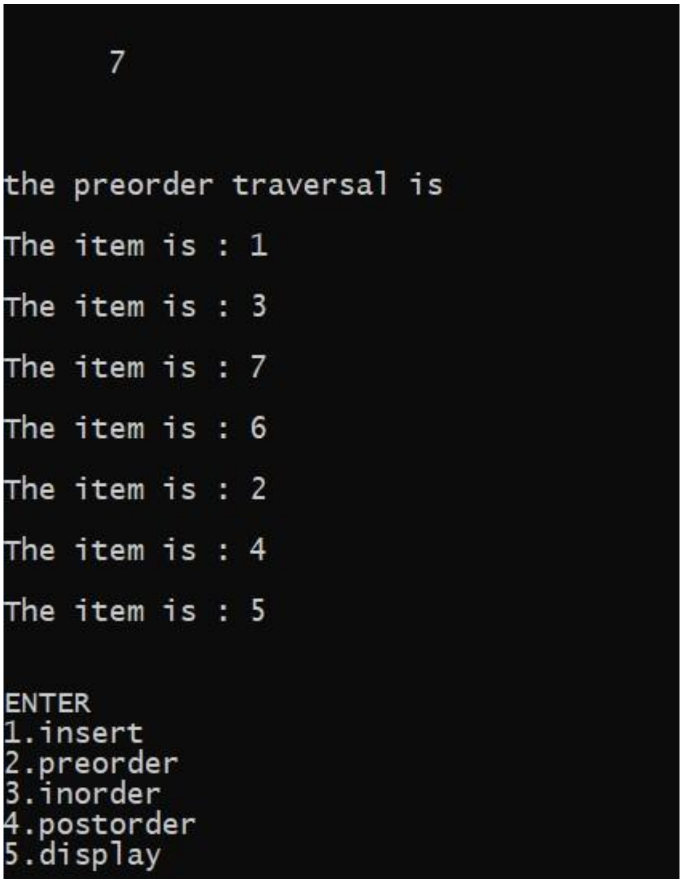
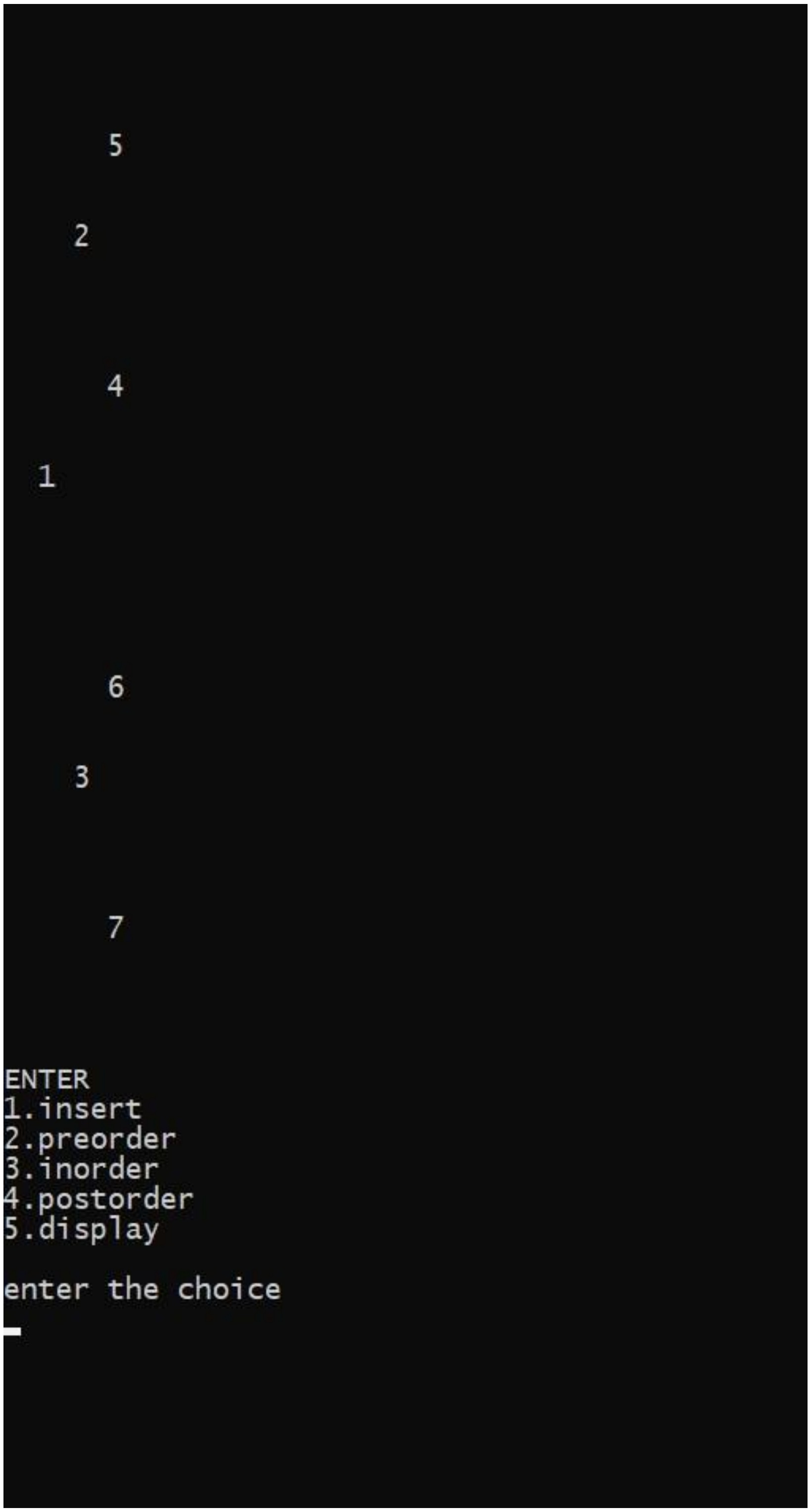
```
#include<stdio.h>
#include<conio.h>


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("\ngive 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("\n***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("\nThe item is : %d\n",root->info);
preorder(root->llink);
preorder(root->rlink);
}
}


void inorder(NODE root)
{
if(root!=NULL)
{
inorder(root->llink);
printf("\nThe item is : %d\n",root->info);
inorder(root->rlink);
}
}



void postorder(NODE root)
{
if (root!=NULL)
{
postorder(root->llink);
postorder(root->rlink);
printf("\nThe item is : %d\n",root->info);
}
```

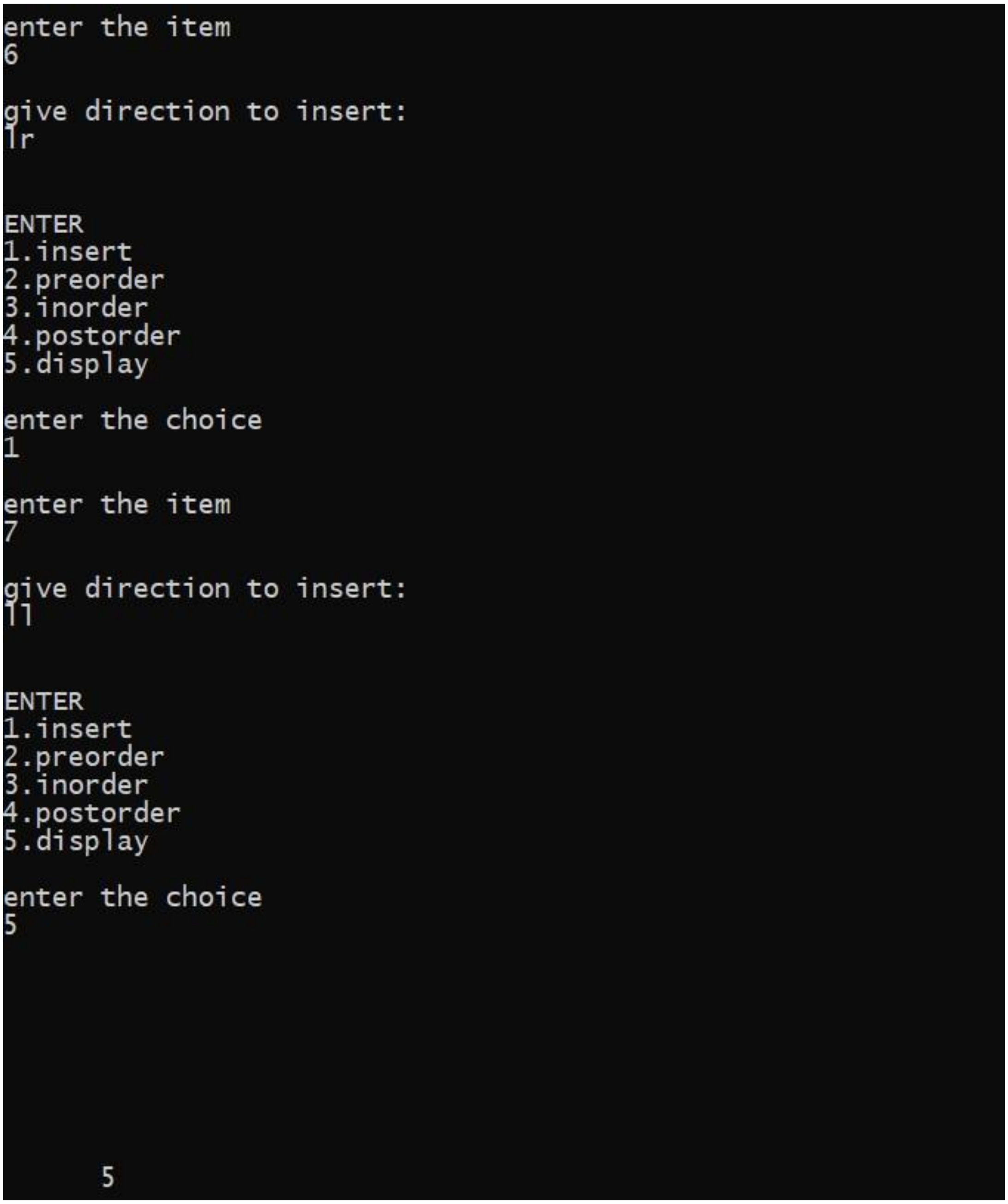
```
}



void display(NODE root,int i)
{
int j;
printf("\n\n");
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("\n\nENTER\n1.insert\n2.preorder\n3.inorder\n4.postorder\n5.dis
play\n");
printf("\nenter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\nenter the item\n");
           scanf("%d",&item);
           root=insert(item,root);
           break;
case 2: if(root==NULL)
           {
            printf("\ntree is empty\n");
           }
           else
           {
            printf("\ngiven tree is\n");
            display(root,1);
            printf("\nthe preorder traversal is \n");
            preorder(root);
           }
           break;
case 3:if(root==NULL)
         {
           printf("\ntree is empty\n");
        }
        else
        {
           printf("\ngiven tree is\n");
           display(root,1);
           printf("\nthe inorder traversal is \n");
           inorder(root);
           }
        break;
case 4:if (root==NULL)
           {
           printf("\ntree is empty\n");
            }
        else
        {
           printf("\ngiven tree is\n");
           display(root,1);
           printf("\nthe postorder traversal is \n");
           postorder(root);
        }
       break;
case 5:display(root,1);
        break;
default:exit(0);
}
}
}
```



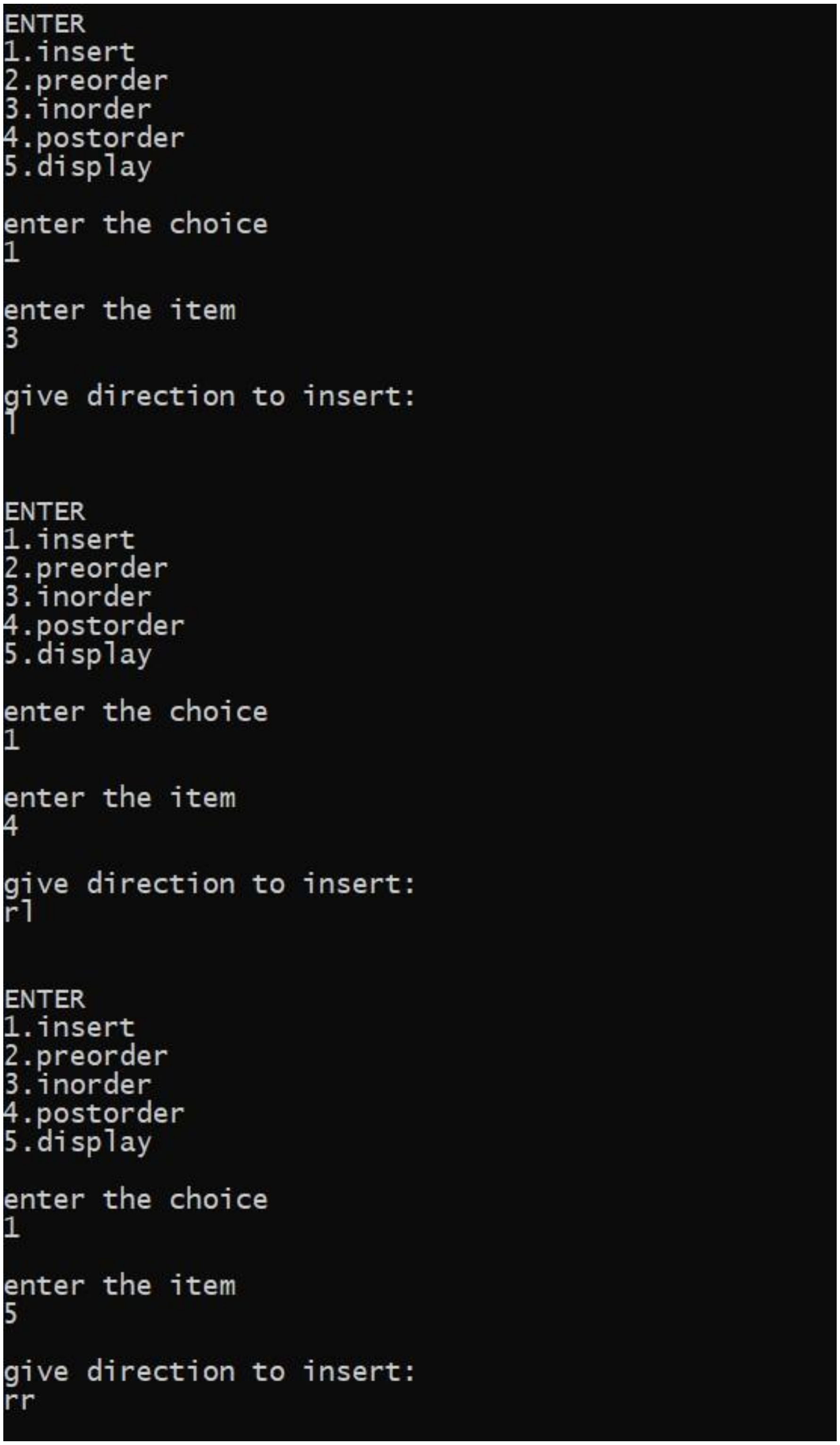
```
      7


the preorder traversal is

The item is : 1

The item is : 3

The item is : 7

The item is : 6

The item is : 2

The item is : 4

The item is : 5


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



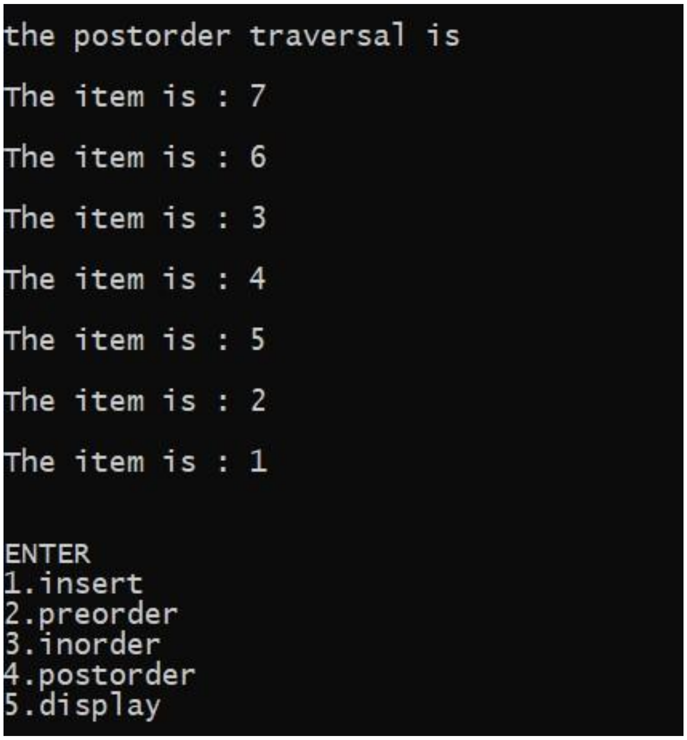
```
      5


    2




      4


  1






      6


    3




      7




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

enter the choice
```



```
enter the item
6

give direction to insert:
lr


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

enter the choice
1

enter the item
7

give direction to insert:
ll


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

enter the choice
5








      5
```



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

enter the choice
1

enter the item
3

give direction to insert:
l


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

enter the choice
1

enter the item
4

give direction to insert:
rl


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

enter the choice
1

enter the item
5

give direction to insert:
rr
```



```
the postorder traversal is

The item is : 7

The item is : 6

The item is : 3

The item is : 4

The item is : 5

The item is : 2

The item is : 1


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