NAME:KUSUM M R DATE:30/12/2020
USN:1BM19CS077
Finding the Maximum element in a tree, Minimum element in a Binary search tree
SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
 {
  int info;
  struct node *rlink;
  struct node *llink;
 };
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
```

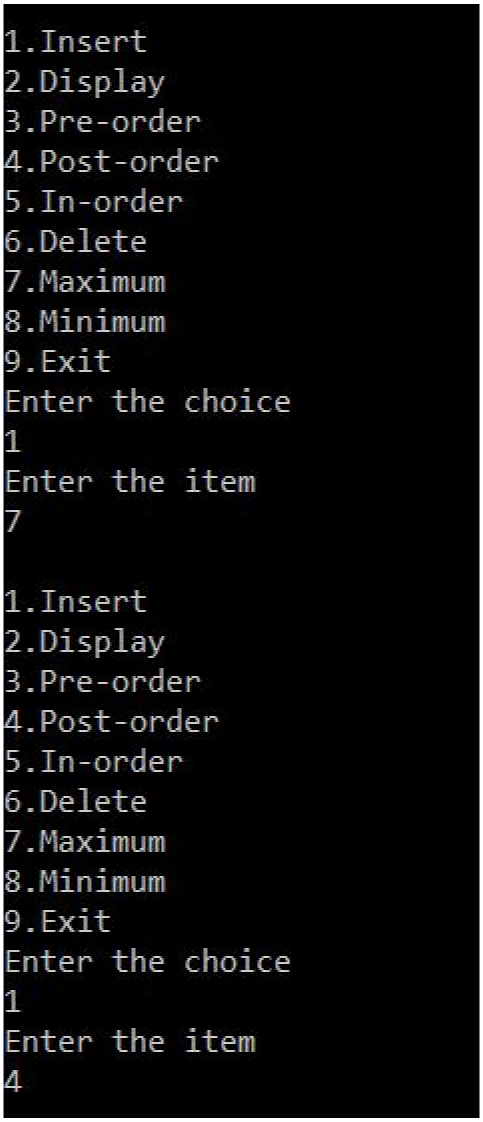
```
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
               printf("  ");
   printf("%d\n",root->info);
              display(root->llink,i+1);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
```

```
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
else
 {
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

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

  postorder(root->llink);
  postorder(root->rlink);
  printf("%d\n",root->info);
  }
 }
void inorder(NODE root)
{
```

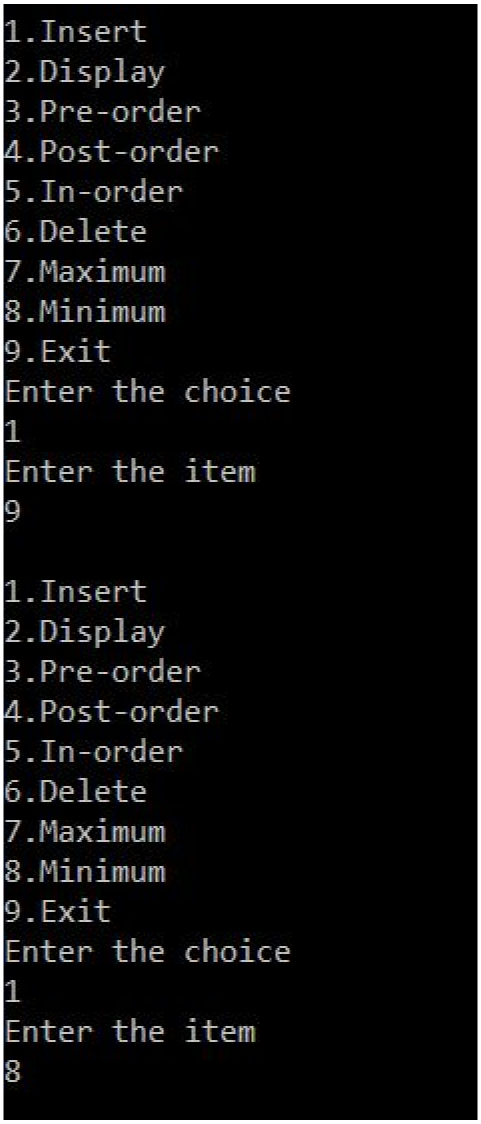
```
if(root!=NULL)
 {

  inorder(root->llink);
  printf("%d\n",root->info);
  inorder(root->rlink);
  }
 }
 void largest(NODE root)
{
    while (root != NULL && root->rlink != NULL)
    {
        root = root->rlink;
    }
    printf("\nLargest value is %d", root->info);
}
void smallest(NODE root)
{
    while (root != NULL && root->llink != NULL)
    {
        root = root->llink;
    }
    printf("\nSmallest value is %d", root->info);
}
void main()
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.Insert\n2.Display\n3.Pre-order\n4.Post-order\n5.In-order\n6.Delete\n7.Maximum\n8
.Minimum\n9.Exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("Enter the item\n");
                           scanf("%d",&item);
                           root=insert(root,item);
                           break;
  case 2:printf("Contents of tree:\n");
      display(root,0);
```

```
                    break;
 case 3:preorder(root);
                    break;
 case 4:postorder(root);
                    break;
 case 5:inorder(root);
                    break;
 case 6:printf("Enter the item\n");
                    scanf("%d",&item);
                    root=delete(root,item);
                    break;
 case 7:largest(root);
                    break;
 case 8:smallest(root);
 break;
 default:exit(0);
                     break;
           }
         }
}
```

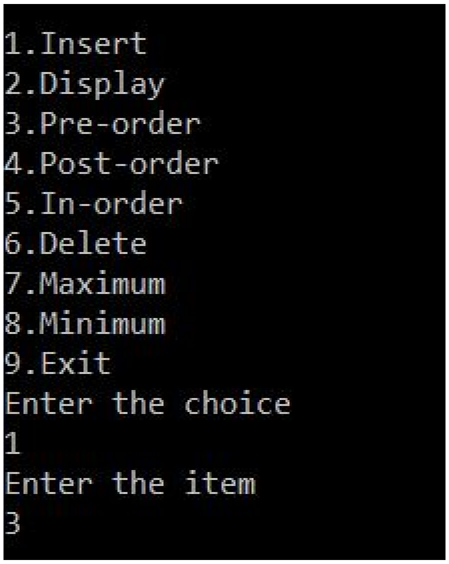
OUTPUT:



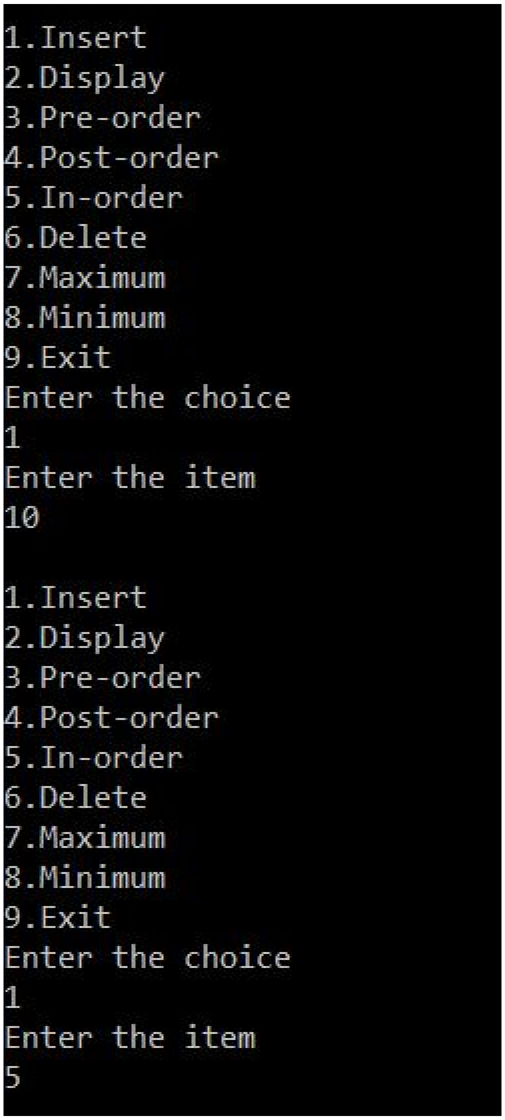
```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
7

1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
4
```



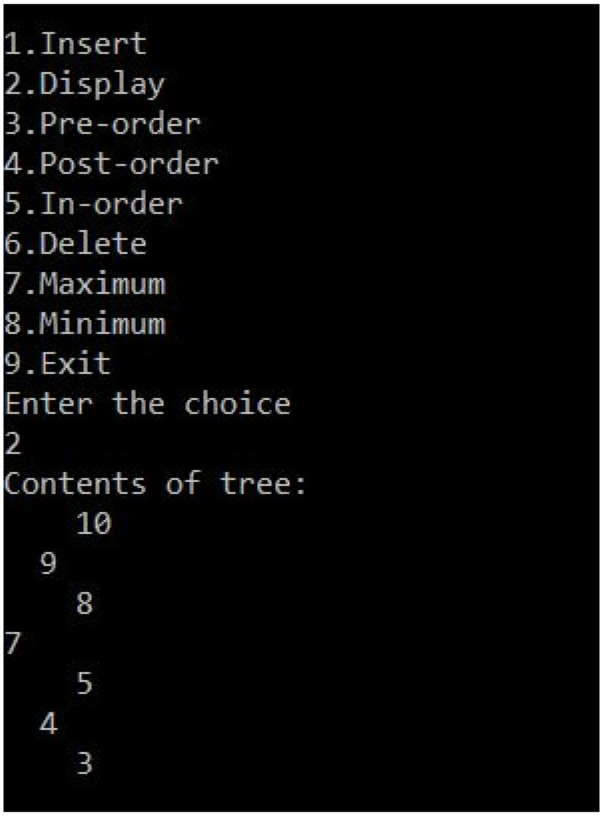
```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
9

1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
8
```



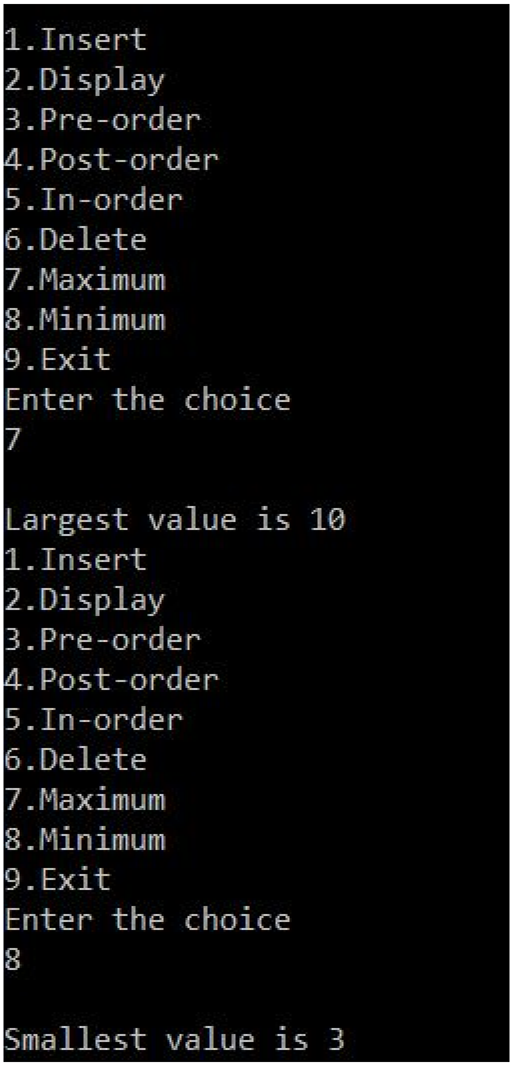
```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
3
```



```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
10

1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
1
Enter the item
5
```



```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
2
Contents of tree:
    10
  9
    8
7
    5
  4
    3
```



```
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
7

Largest value is 10
1.Insert
2.Display
3.Pre-order
4.Post-order
5.In-order
6.Delete
7.Maximum
8.Minimum
9.Exit
Enter the choice
8

Smallest value is 3
```

Binary tree program(Count the number of nodes in a tree)

SOURCE CODE:

```
#include<stdio.h>
struct node
{
int info;
struct node*llink;
struct node*rlink;
};
typedef struct node*NODE;
NODE getnode()
{
```

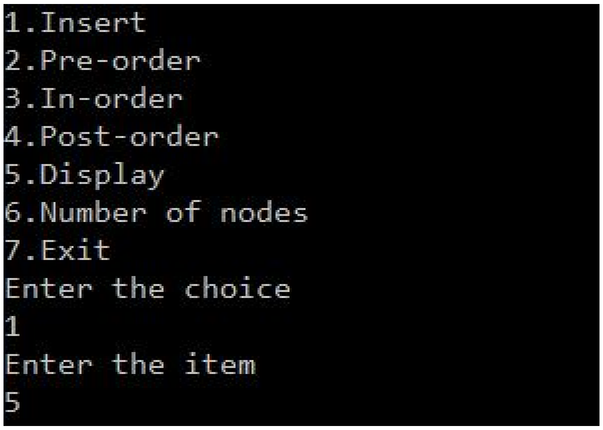
```
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("Memory not available");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(int item,NODE root)
{
NODE temp,cur,prev;
char direction[10];
int i;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;
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);
}
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)
```

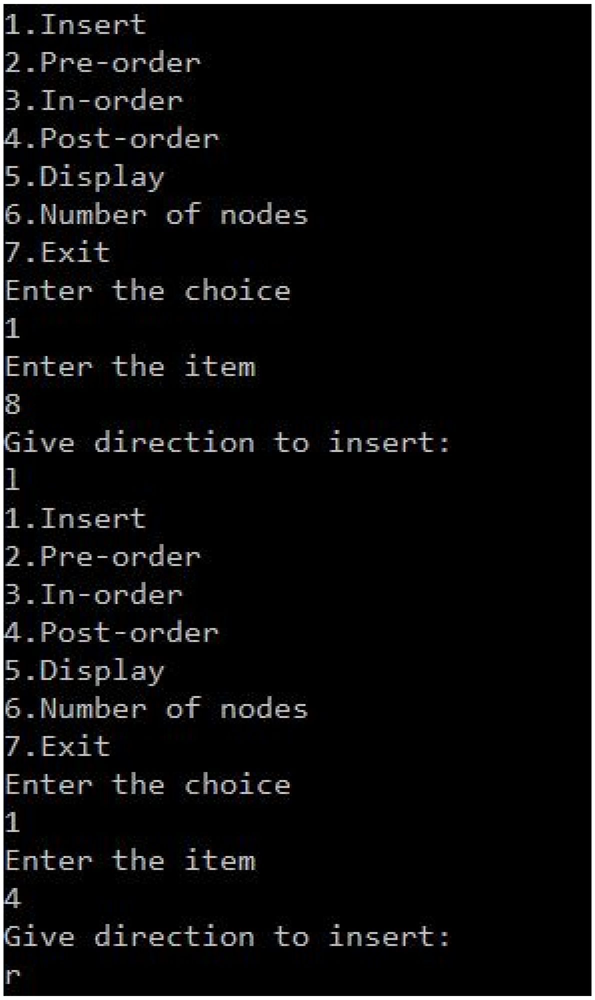
```
{
postorder(root->llink);
postorder(root->rlink);
printf("The item is %d\n",root->info);
}
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
{
display(root->rlink,i+1);
for (j=1;j<=i;j++)
printf("  ");
printf("%d\n",root->info);
display(root->llink,i+1);
}
}
int count(NODE root)
{
   int c=1;
   if (root ==NULL)
      return 0;

   else
   {
      c += count(root->llink);
      c += count(root->rlink);
      return c;
   }
}

void main()
{
NODE root=NULL;
int choice,i,item;
```

```
for(;;)
{
printf("1.Insert\n2.Pre-order\n3.In-order\n4.Post-order\n5.Display\n6.Number of nodes\n7.Exit\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 pre-order 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 in-order 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:printf("Contents of tree:\n");
   display(root,1);
          break;
          case 6:
          printf("Number of nodes: %d\n",count(root));
          break;
default:exit(0);
}
}
}
```

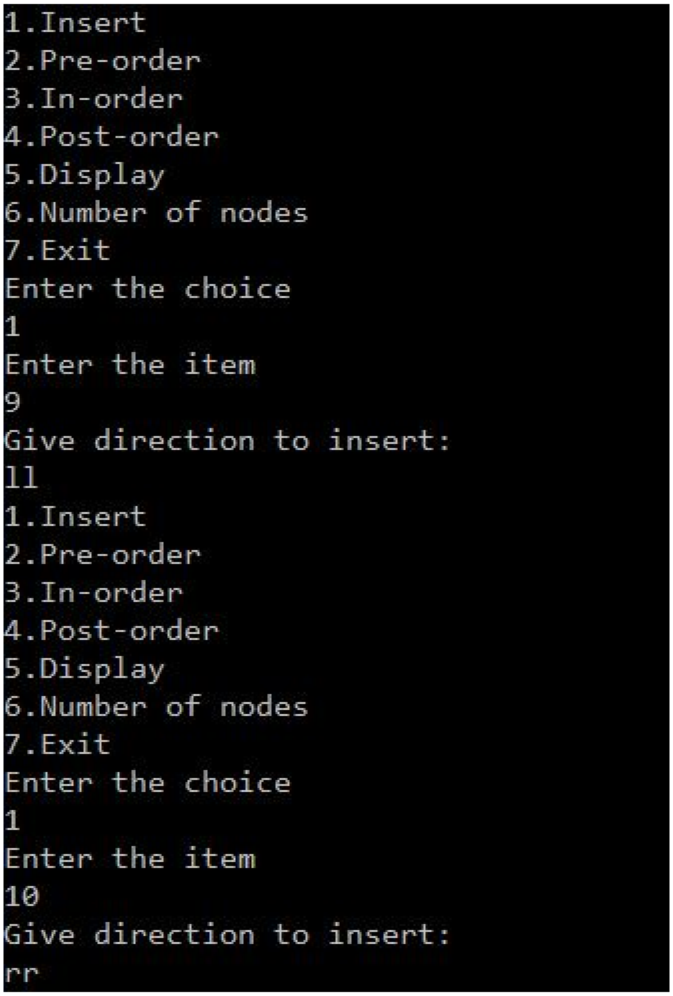
OUTPUT:



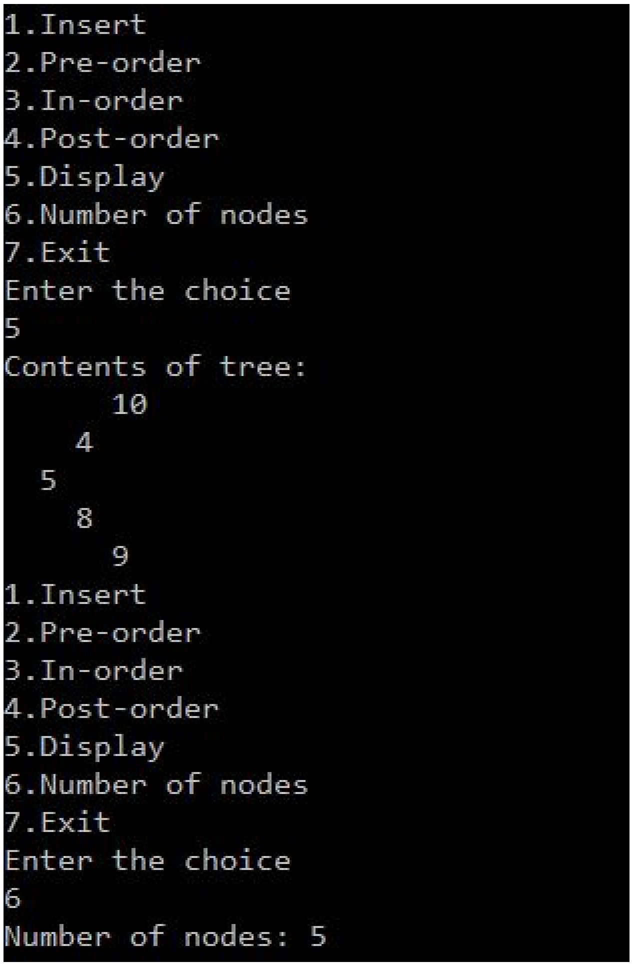
```
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
1
Enter the item
5
```



```
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
1
Enter the item
8
Give direction to insert:
l
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
1
Enter the item
4
Give direction to insert:
r
```



```
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
1
Enter the item
9
Give direction to insert:
ll
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
1
Enter the item
10
Give direction to insert:
rr
```



```
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
5
Contents of tree:
      10
    4
  5
    8
      9
1.Insert
2.Pre-order
3.In-order
4.Post-order
5.Display
6.Number of nodes
7.Exit
Enter the choice
6
Number of nodes: 5
```