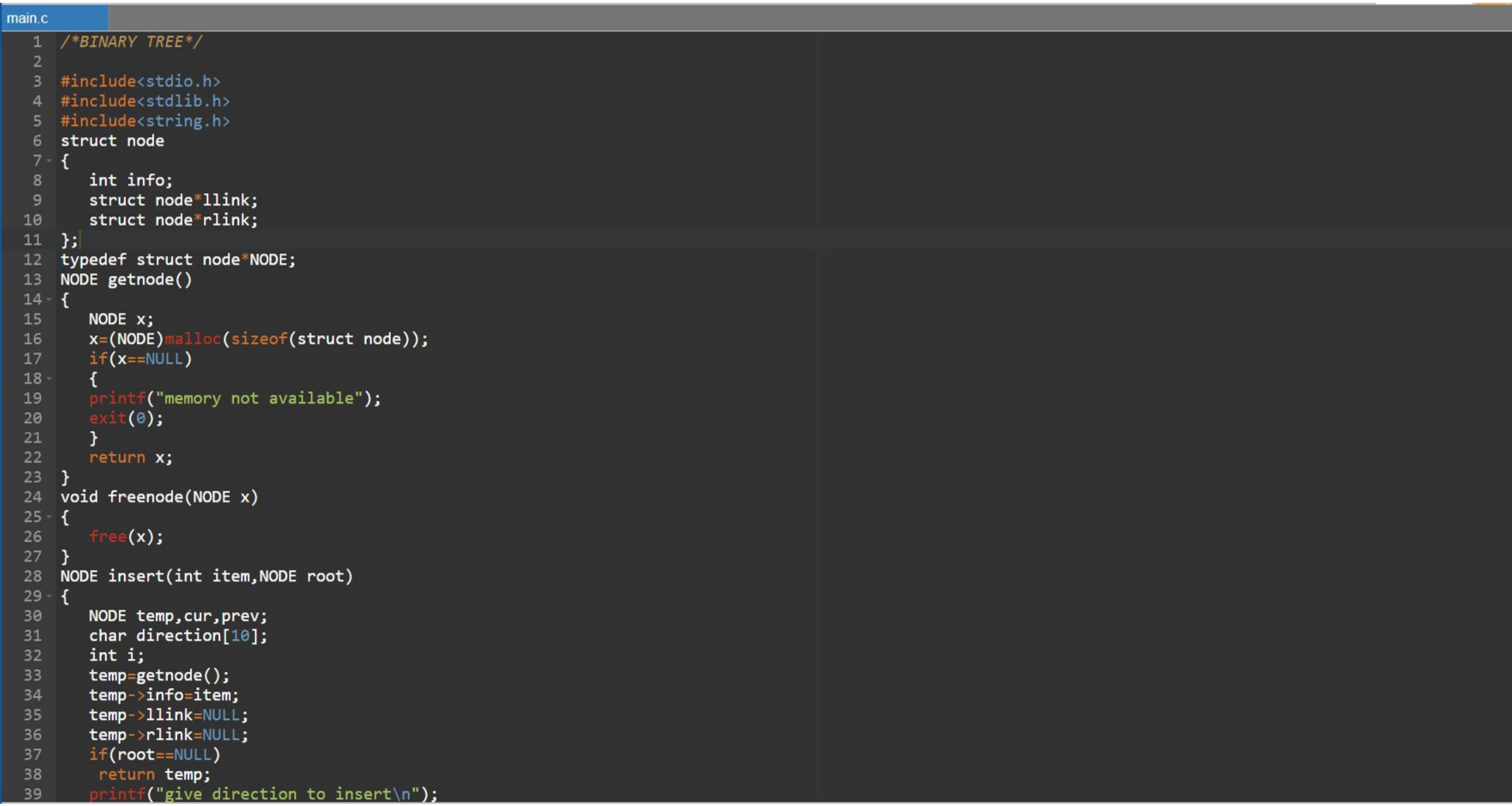
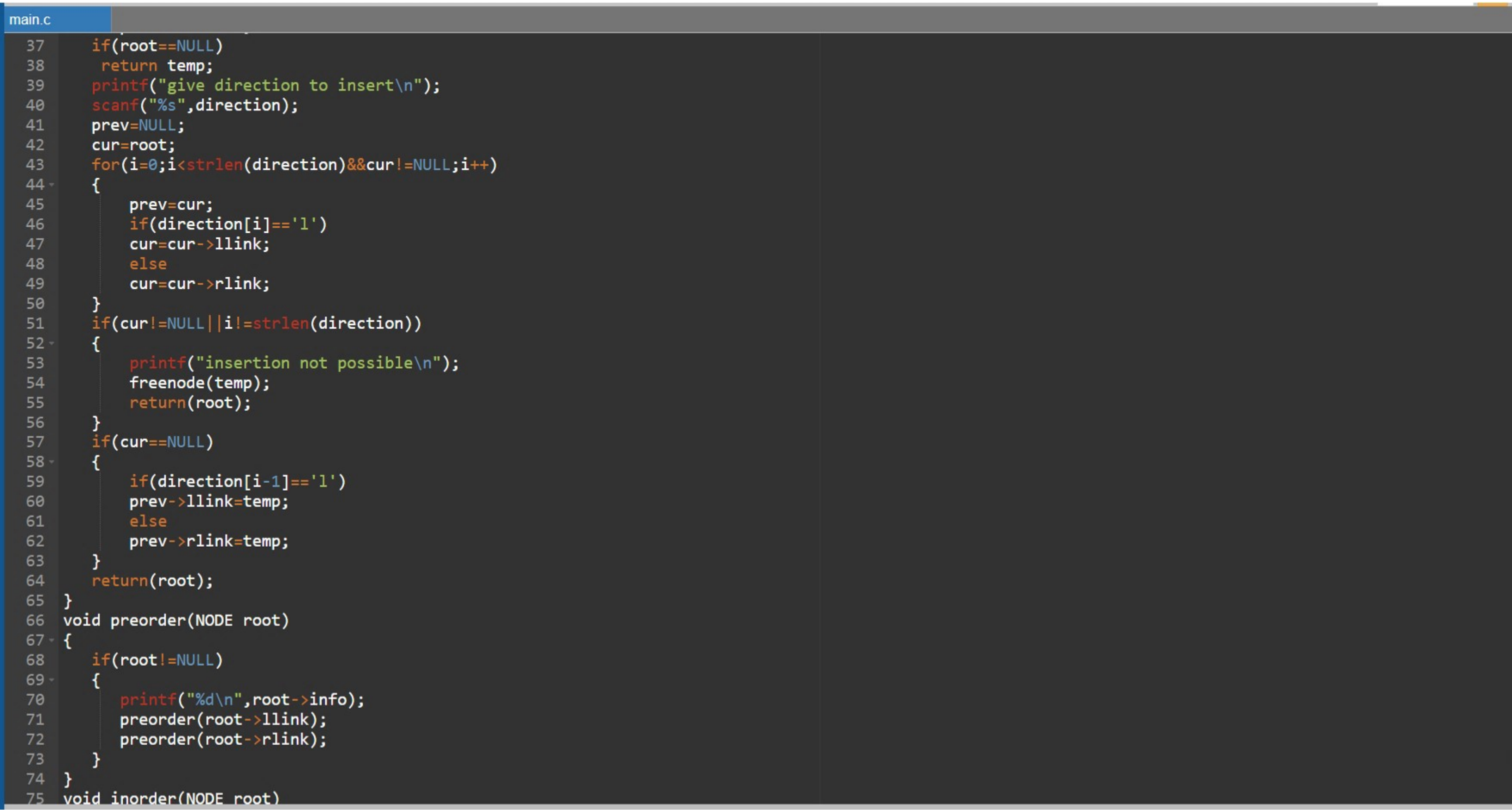
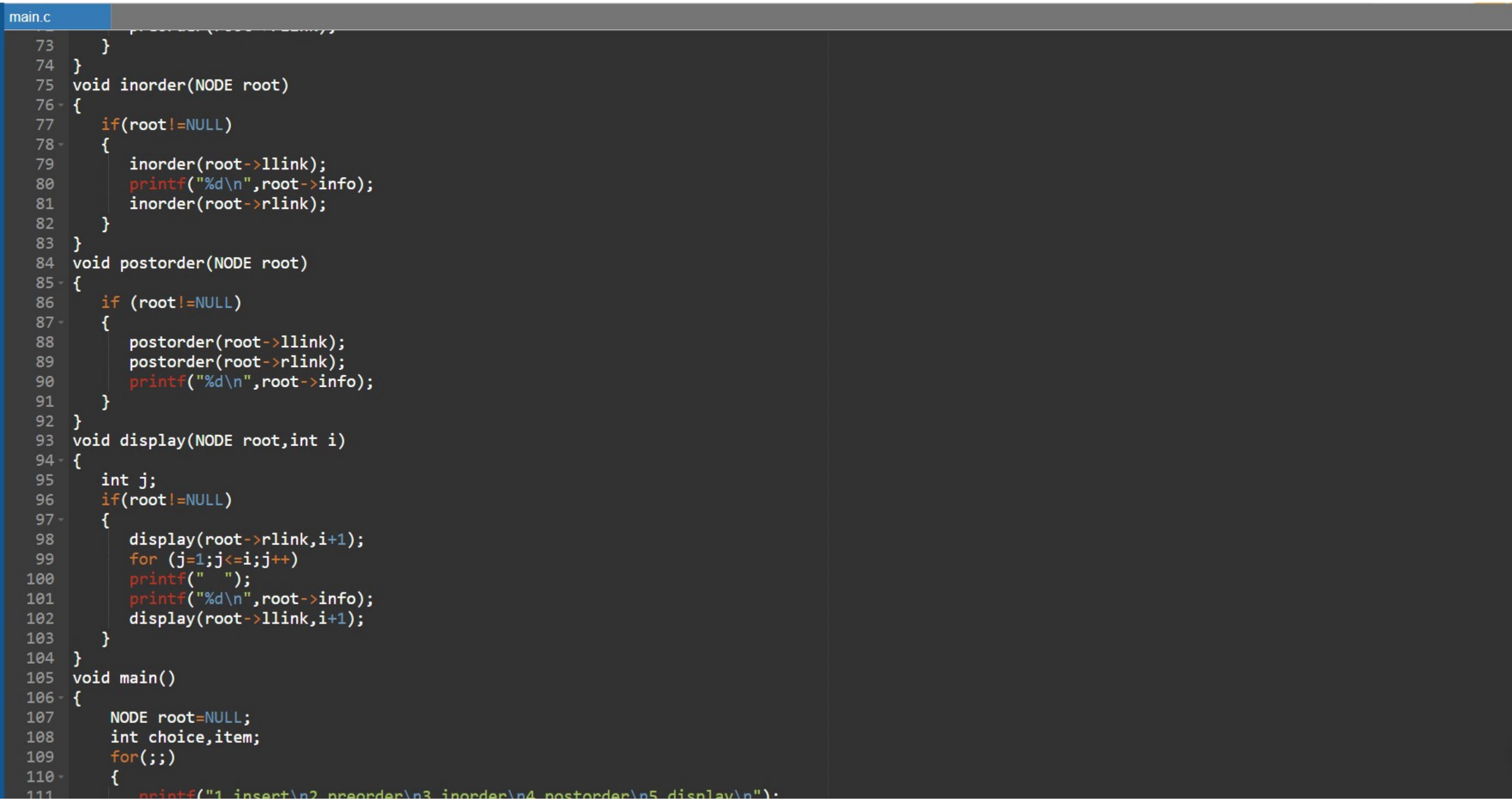

main.c

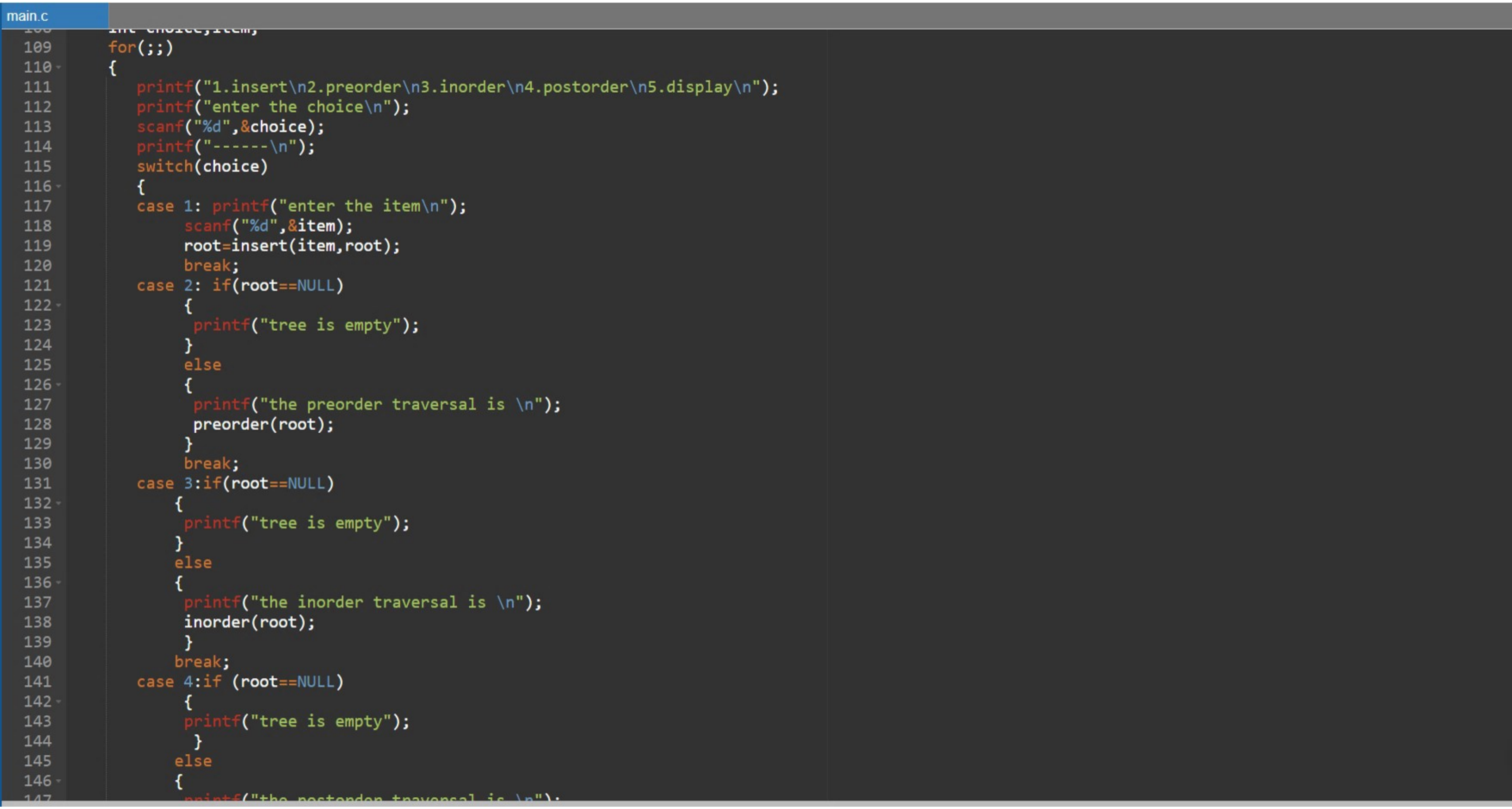
```c
/*BINARY TREE*/

#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");
```


main.c

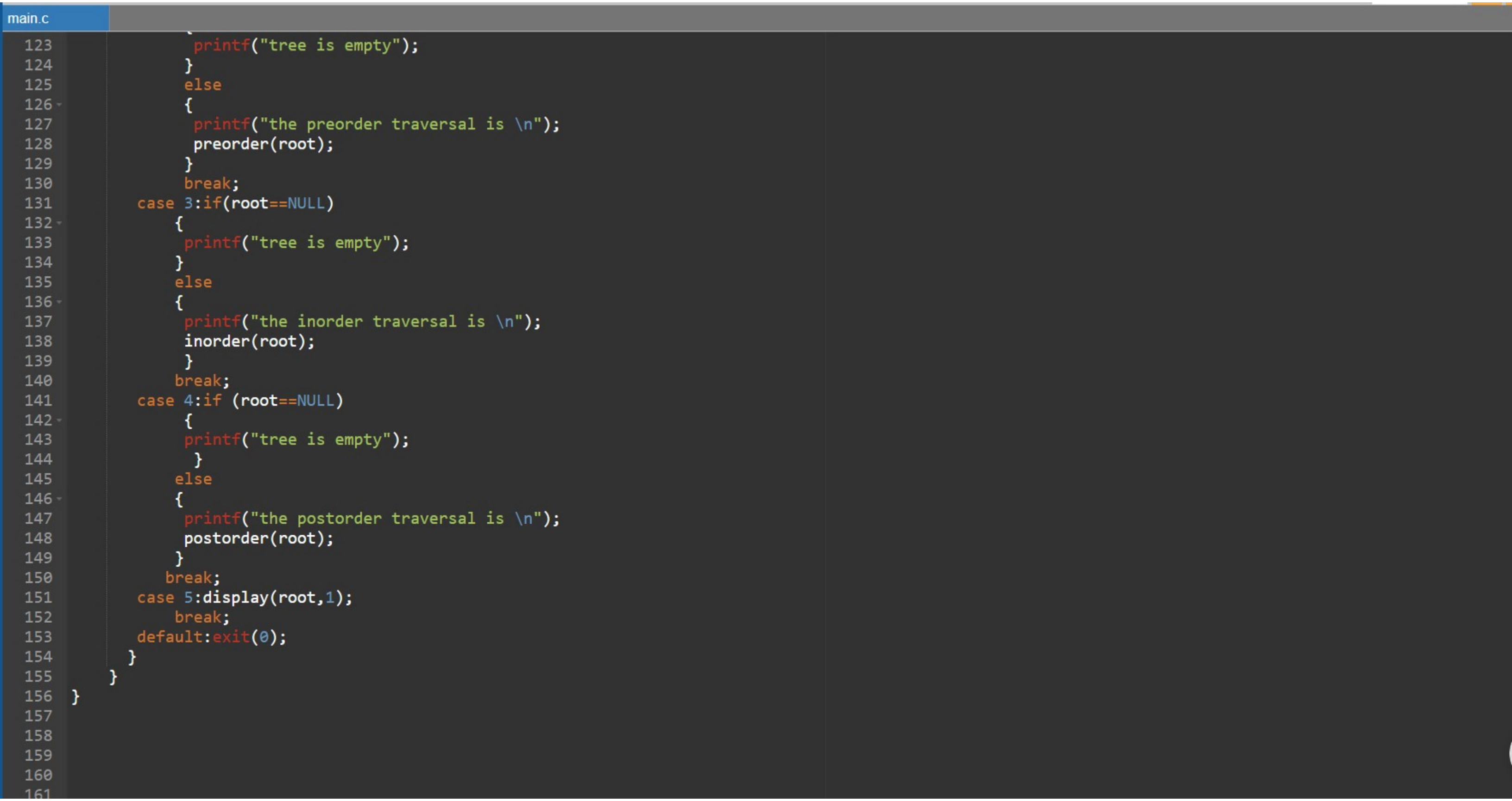
```c
    if(root==NULL)
     return temp;
    printf("give direction to insert\n");
    scanf("%s",direction);
    prev=NULL;
    cur=root;
    for(i=0;i<strlen(direction)&&cur!=NULL;i++)
    {
        prev=cur;
        if(direction[i]=='l')
        cur=cur->llink;
        else
        cur=cur->rlink;
    }
    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("%d\n",root->info);
       preorder(root->llink);
       preorder(root->rlink);
    }
}
void inorder(NODE root)
```


main.c

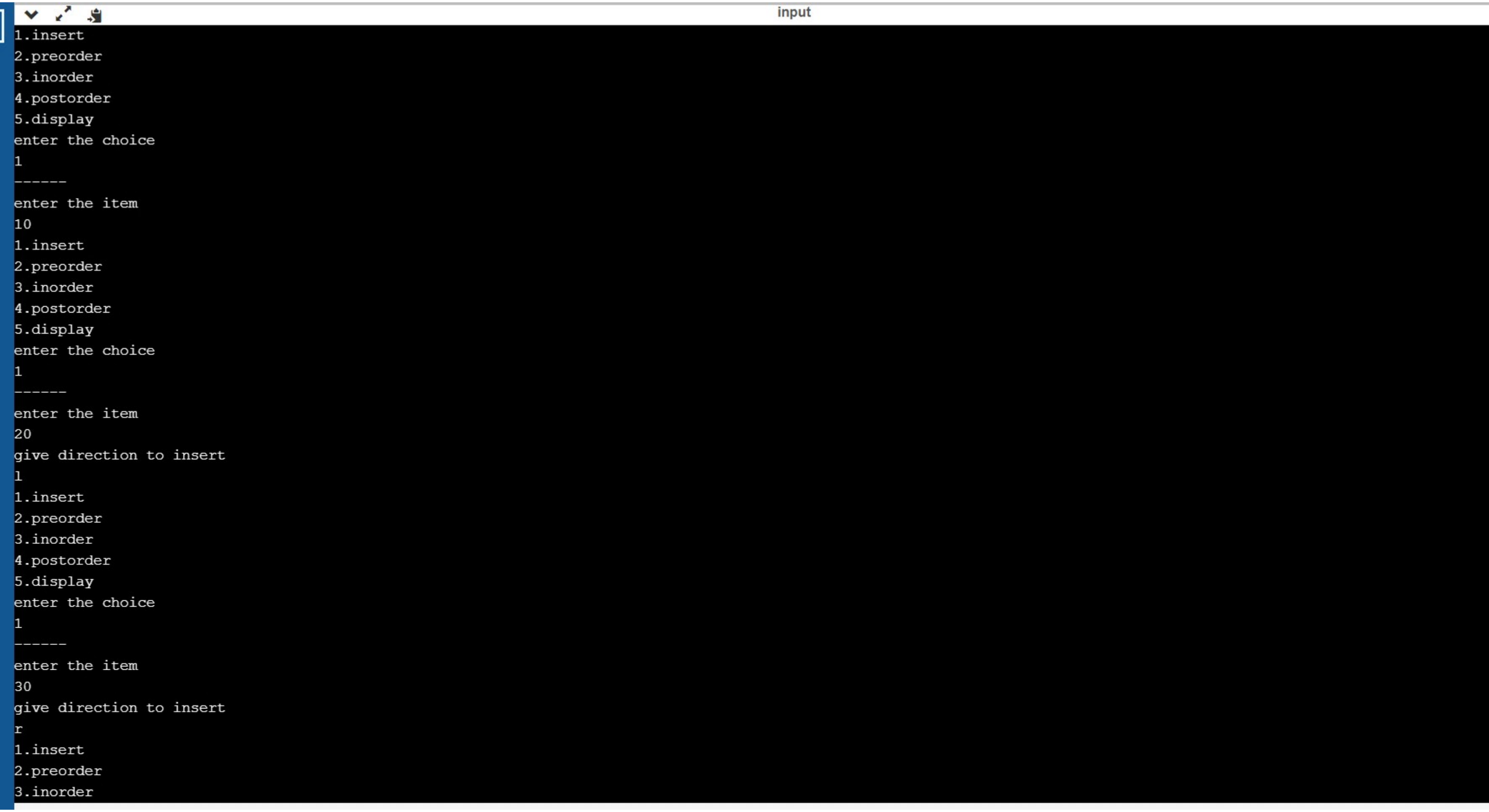
```c
   }
}
void inorder(NODE root)
{
   if(root!=NULL)
   {
      inorder(root->llink);
      printf("%d\n",root->info);
      inorder(root->rlink);
   }
}
void postorder(NODE root)
{
   if (root!=NULL)
   {
      postorder(root->llink);
      postorder(root->rlink);
      printf("%d\n",root->info);
   }
}
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,item;
    for(;;)
    {
       printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n5.display\n");
```


main.c

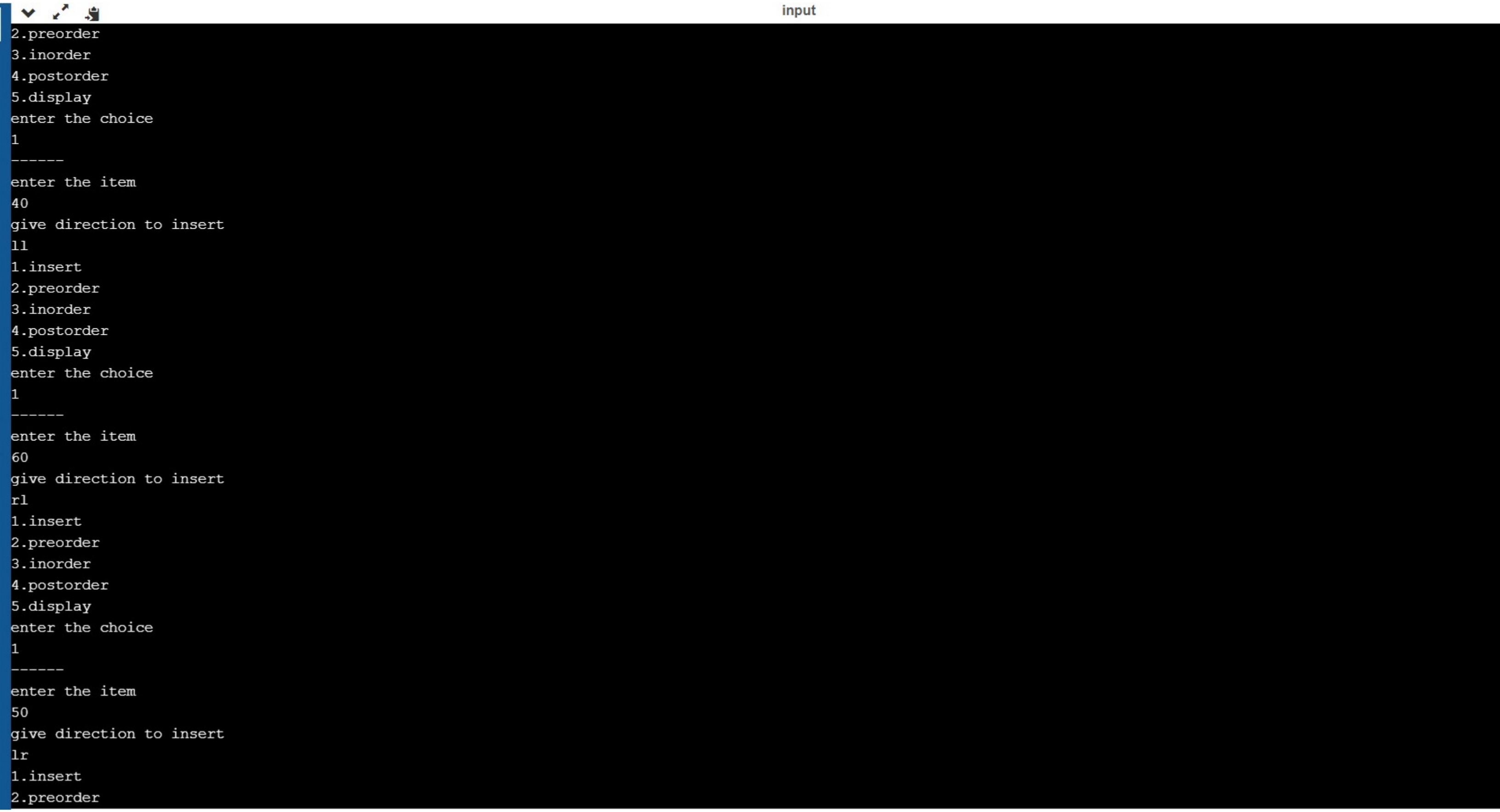
```c
    int choice,item;
    for(;;)
    {
       printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n5.display\n");
       printf("enter the choice\n");
       scanf("%d",&choice);
       printf("------\n");
       switch(choice)
       {
       case 1: printf("enter the item\n");
            scanf("%d",&item);
            root=insert(item,root);
            break;
       case 2: if(root==NULL)
            {
             printf("tree is empty");
            }
            else
            {
             printf("the preorder traversal is \n");
             preorder(root);
            }
            break;
       case 3:if(root==NULL)
           {
            printf("tree is empty");
           }
           else
           {
            printf("the inorder traversal is \n");
            inorder(root);
            }
           break;
       case 4:if (root==NULL)
            {
            printf("tree is empty");
             }
           else
           {
            printf("the postorder traversal is \n");
```



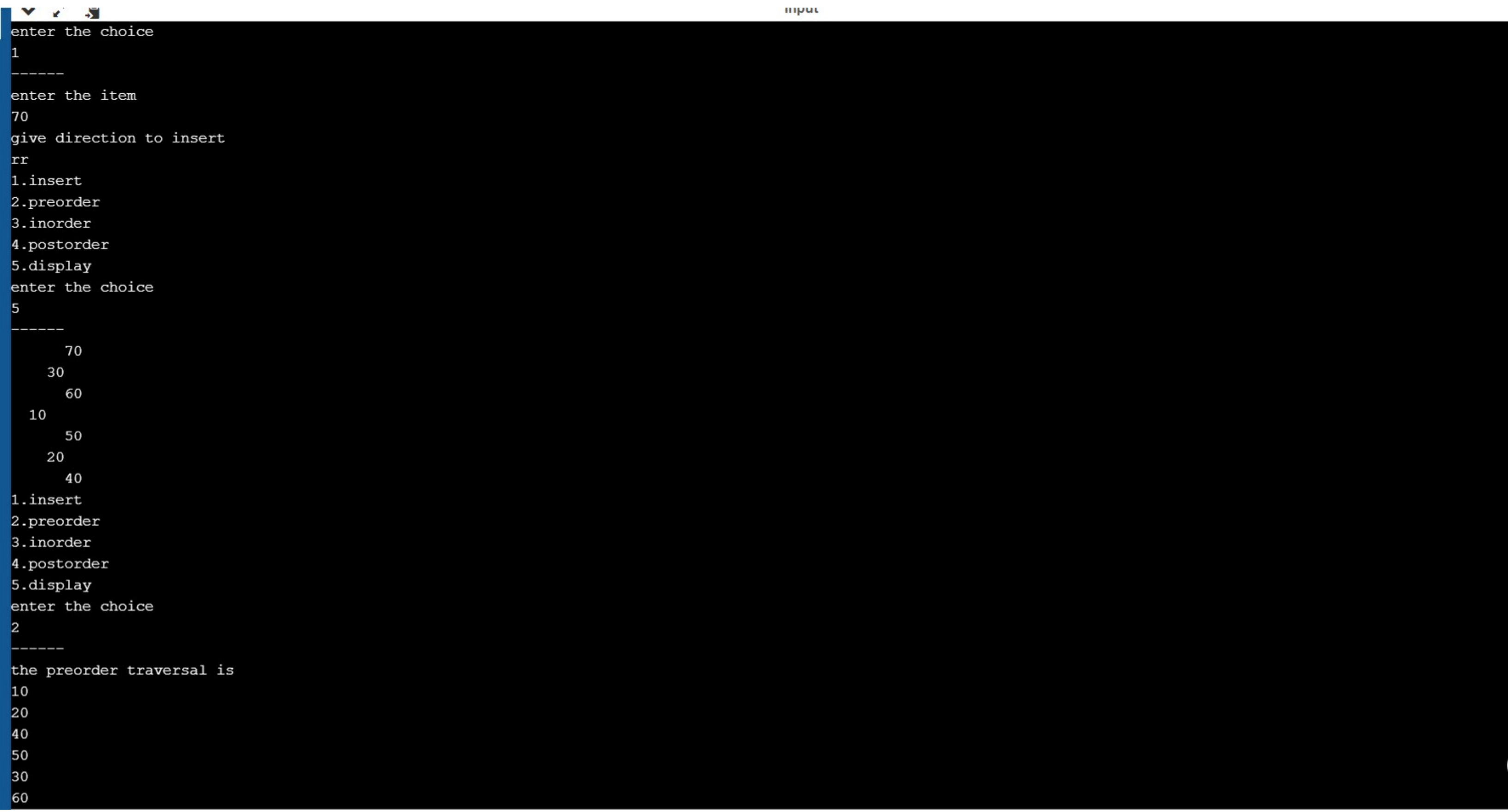
```c
             printf("tree is empty");
            }
            else
            {
             printf("the preorder traversal is \n");
             preorder(root);
            }
            break;
       case 3:if(root==NULL)
           {
            printf("tree is empty");
           }
           else
           {
            printf("the inorder traversal is \n");
            inorder(root);
            }
           break;
       case 4:if (root==NULL)
            {
            printf("tree is empty");
             }
           else
           {
            printf("the postorder traversal is \n");
            postorder(root);
           }
          break;
       case 5:display(root,1);
           break;
       default:exit(0);
      }
    }
}
```


input

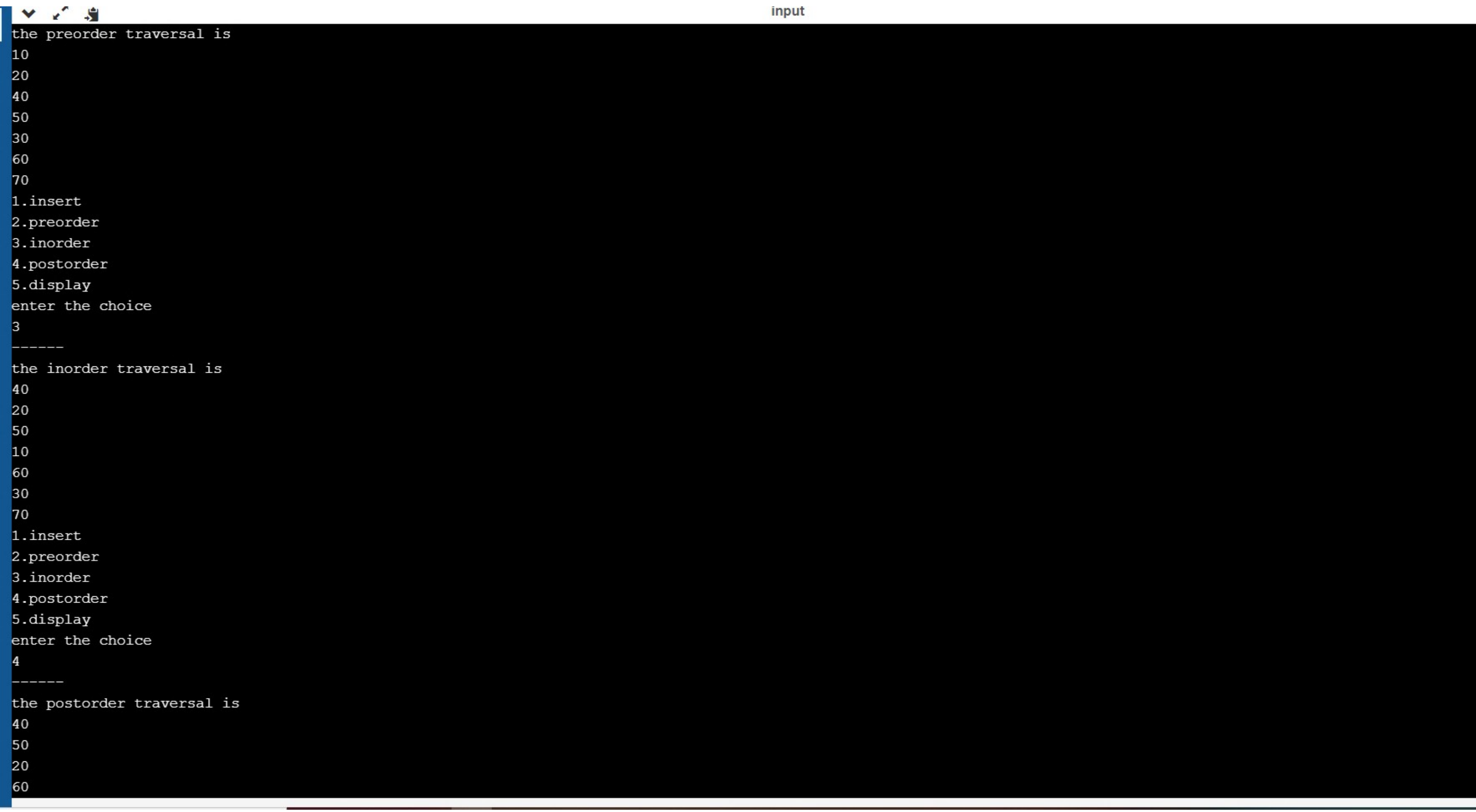
```
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
------
enter the item
10
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
------
enter the item
20
give direction to insert
l
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
------
enter the item
30
give direction to insert
r
1.insert
2.preorder
3.inorder
```


input

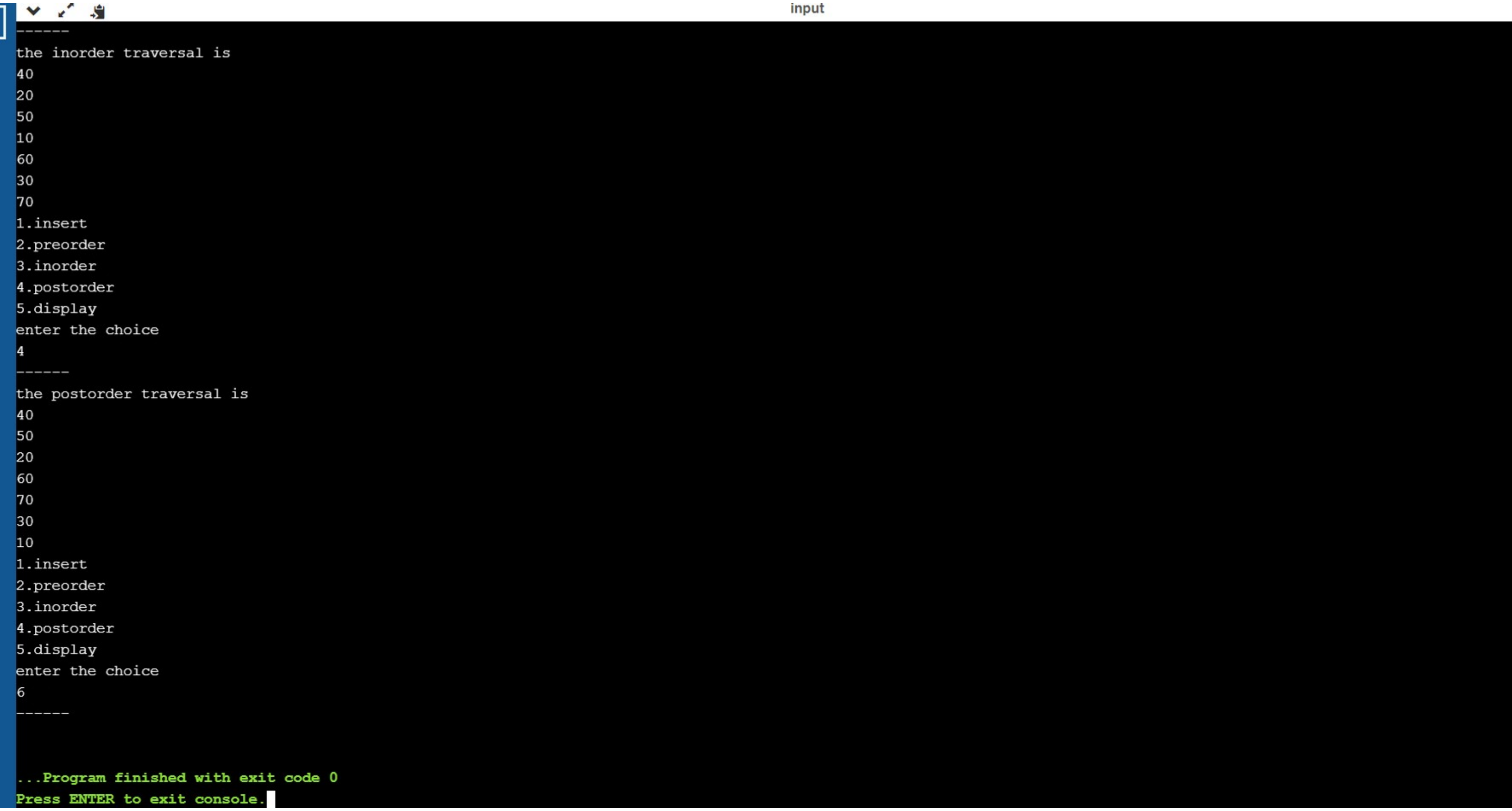
```
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
------
enter the item
40
give direction to insert
ll
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
------
enter the item
60
give direction to insert
rl
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
------
enter the item
50
give direction to insert
lr
1.insert
2.preorder
```


input

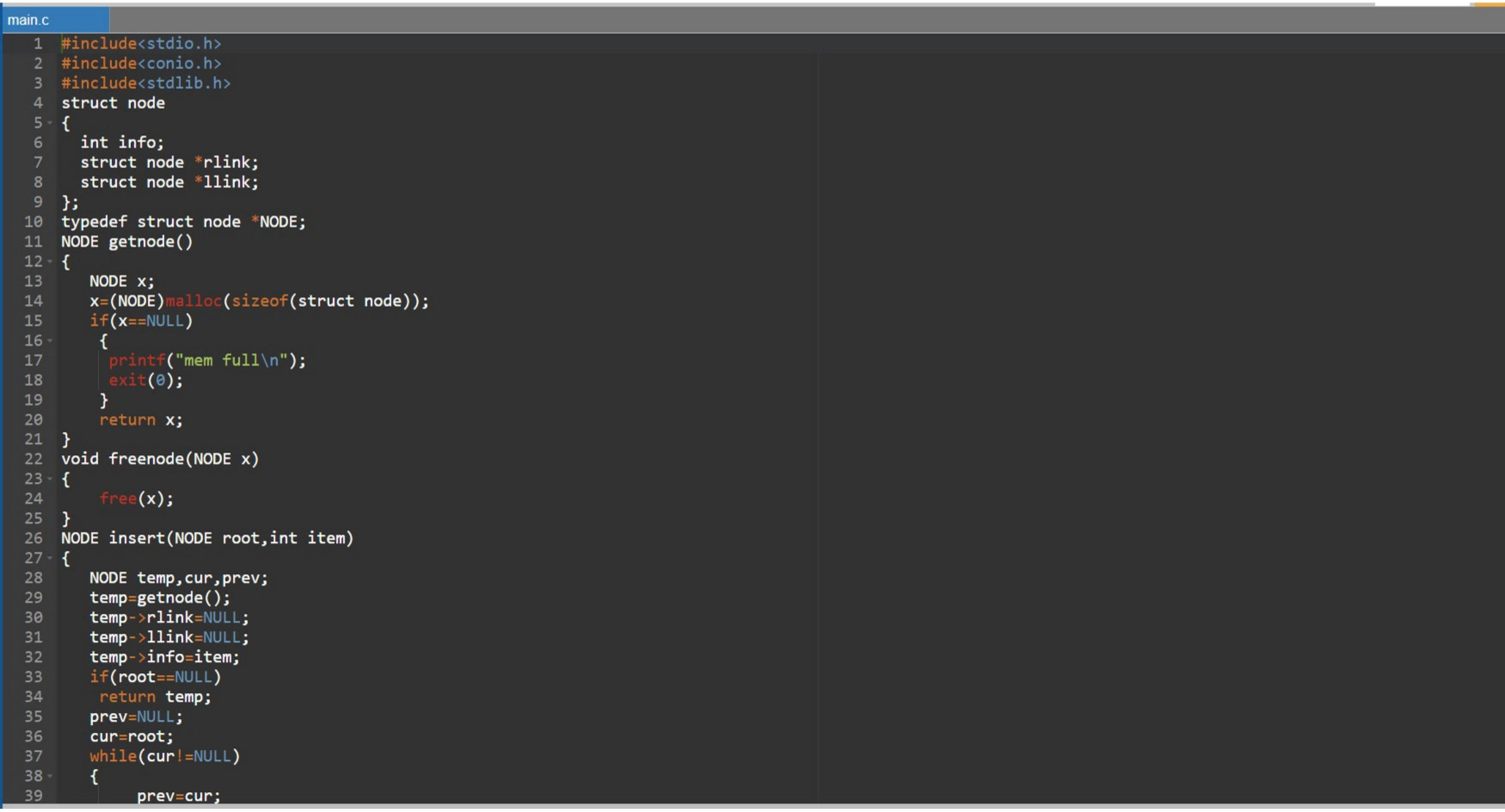
```
enter the choice
1
------
enter the item
70
give direction to insert
rr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
------
      70
    30
      60
  10
      50
    20
      40
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
------
the preorder traversal is
10
20
40
50
30
60
```



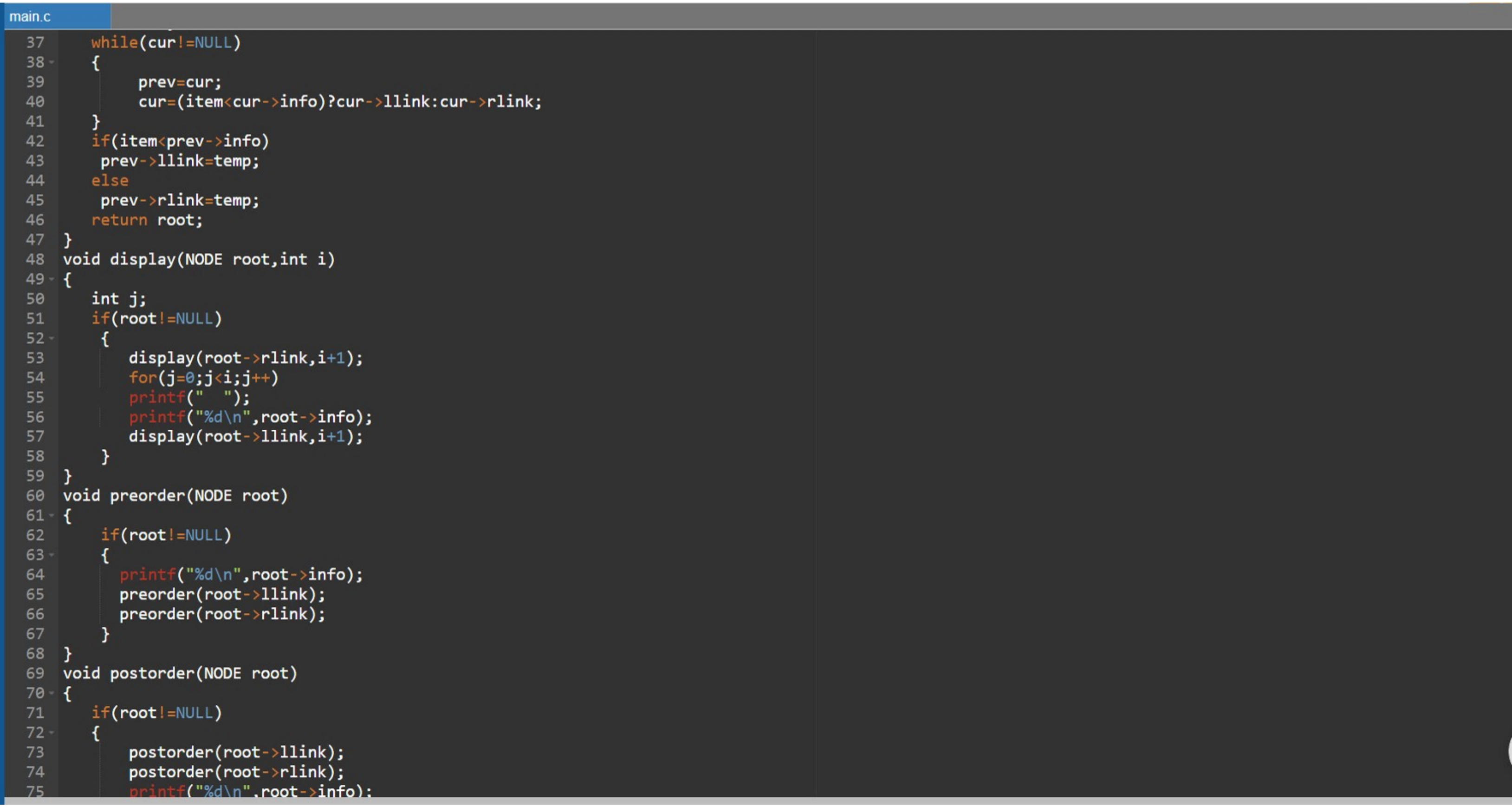
```
the preorder traversal is
10
20
40
50
30
60
70
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
------
the inorder traversal is
40
20
50
10
60
30
70
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
------
the postorder traversal is
40
50
20
60
```


input

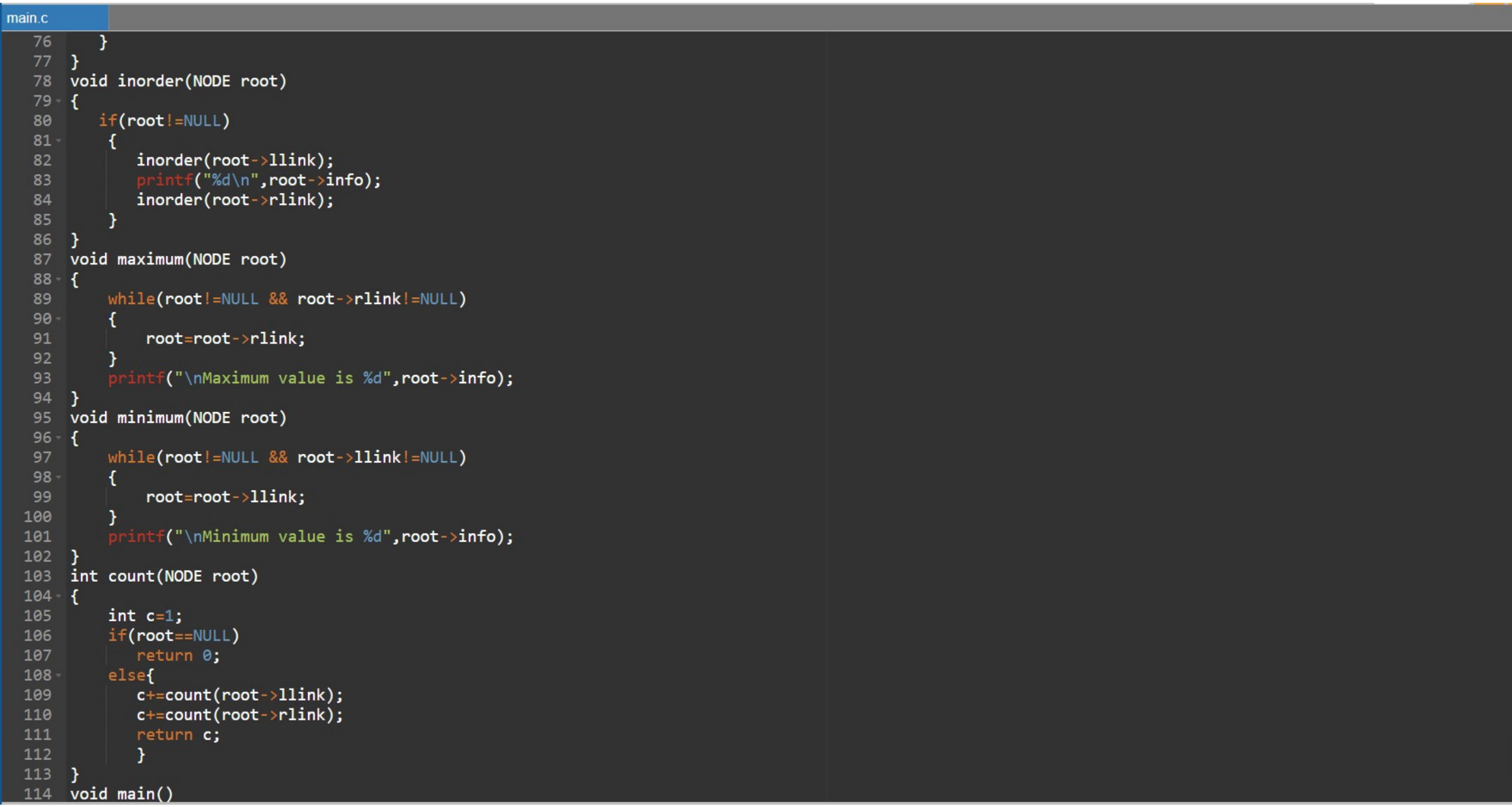
```
------
the inorder traversal is
40
20
50
10
60
30
70
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
------
the postorder traversal is
40
50
20
60
70
30
10
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
6
------


...Program finished with exit code 0
Press ENTER to exit console.
```


main.c

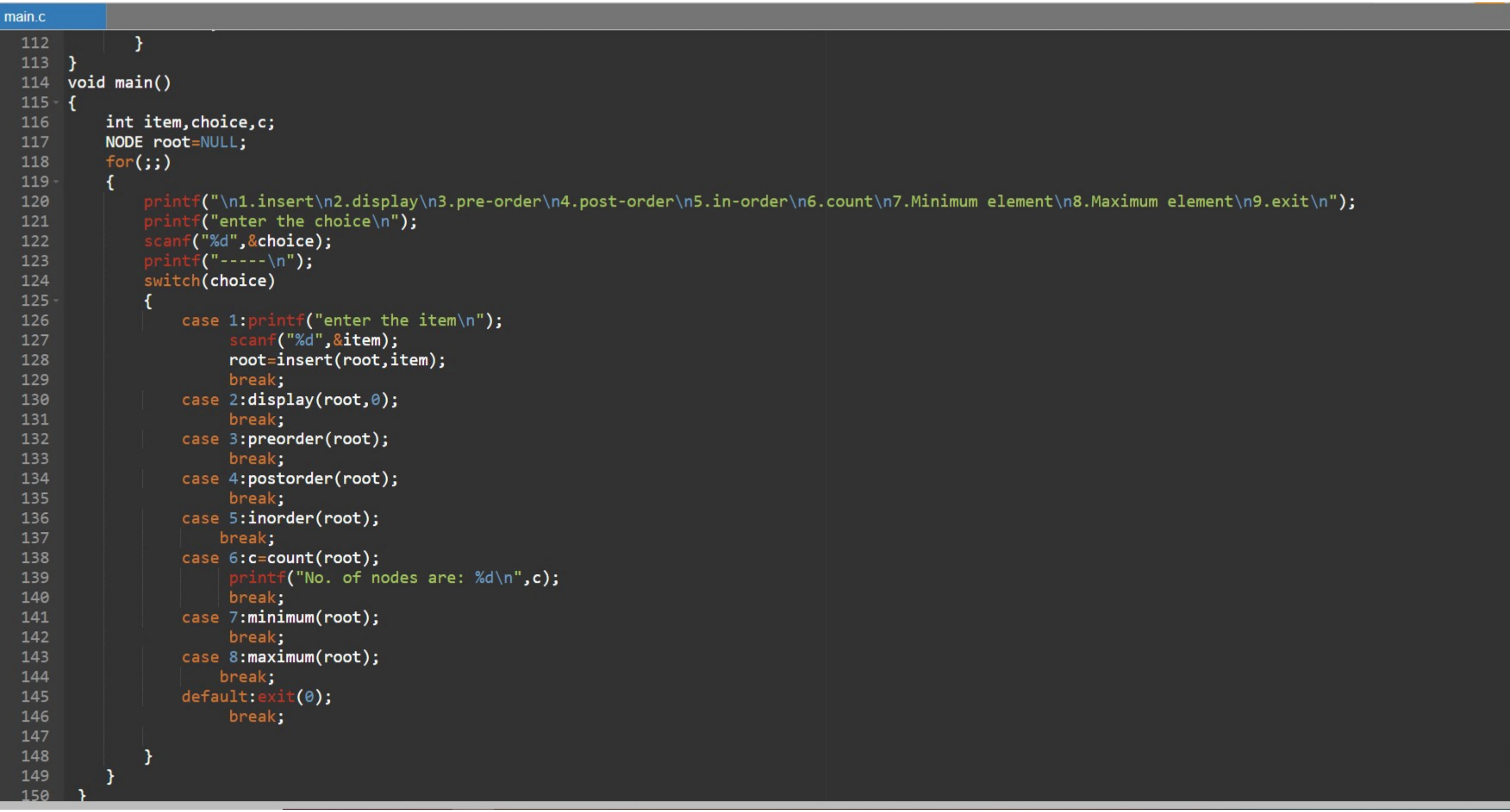
```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node
{
  int info;
  struct node *rlink;
  struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
   NODE x;
   x=(NODE)malloc(sizeof(struct node));
   if(x==NULL)
    {
     printf("mem full\n");
     exit(0);
    }
    return 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;
```


main.c

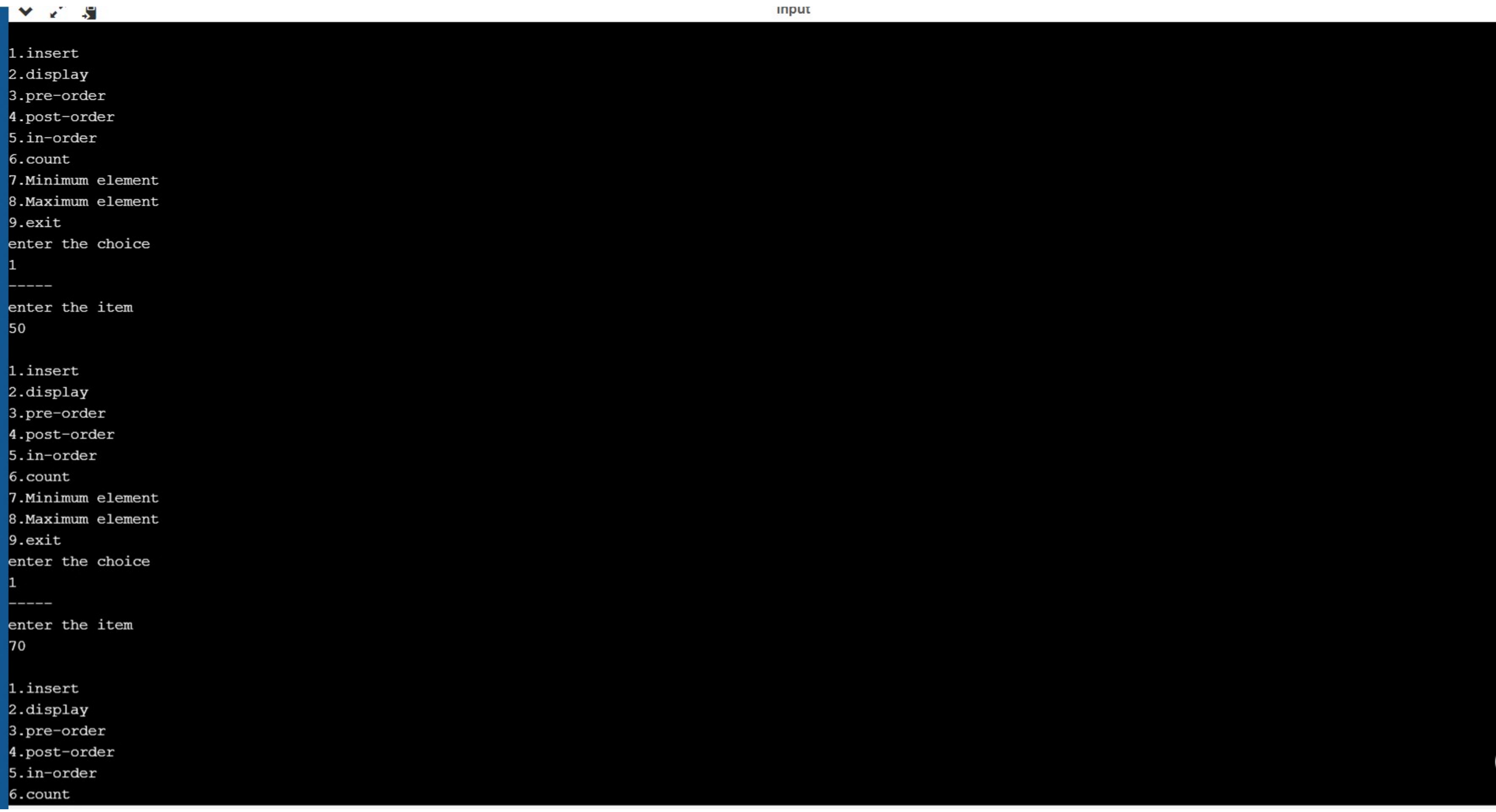
```
    while(cur!=NULL)
    {
         prev=cur;
         cur=(item<cur->info)?cur->llink:cur->rlink;
    }
    if(item<prev->info)
     prev->llink=temp;
    else
     prev->rlink=temp;
    return root;
}
void display(NODE root,int i)
{
    int j;
    if(root!=NULL)
     {
        display(root->rlink,i+1);
        for(j=0;j<i;j++)
        printf("  ");
        printf("%d\n",root->info);
        display(root->llink,i+1);
     }
}
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);
```


main.c

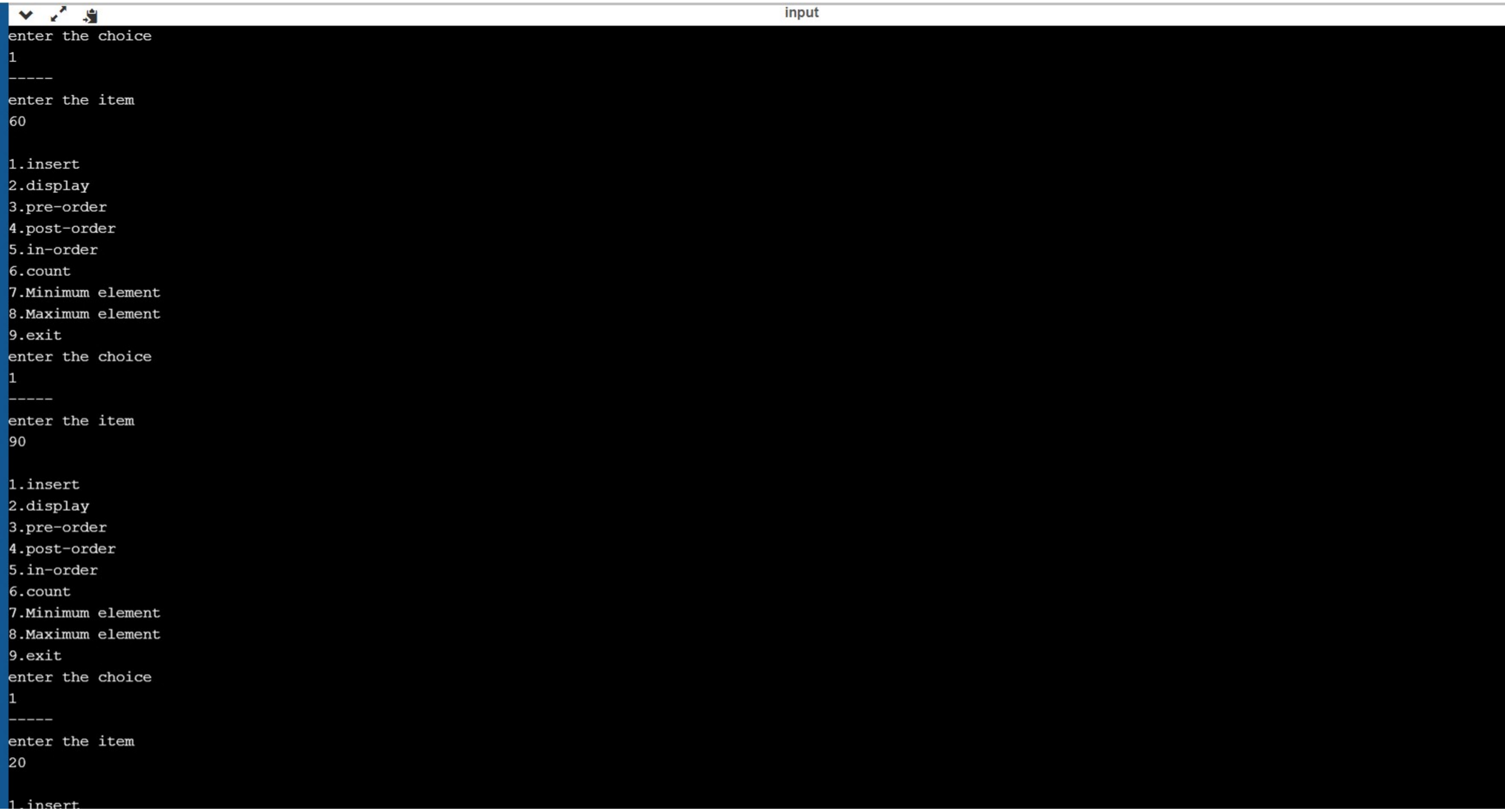
```c
    }
}
void inorder(NODE root)
{
   if(root!=NULL)
    {
       inorder(root->llink);
       printf("%d\n",root->info);
       inorder(root->rlink);
    }
}
void maximum(NODE root)
{
    while(root!=NULL && root->rlink!=NULL)
    {
        root=root->rlink;
    }
    printf("\nMaximum value is %d",root->info);
}
void minimum(NODE root)
{
    while(root!=NULL && root->llink!=NULL)
    {
        root=root->llink;
    }
    printf("\nMinimum value is %d",root->info);
}
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()
```


main.c

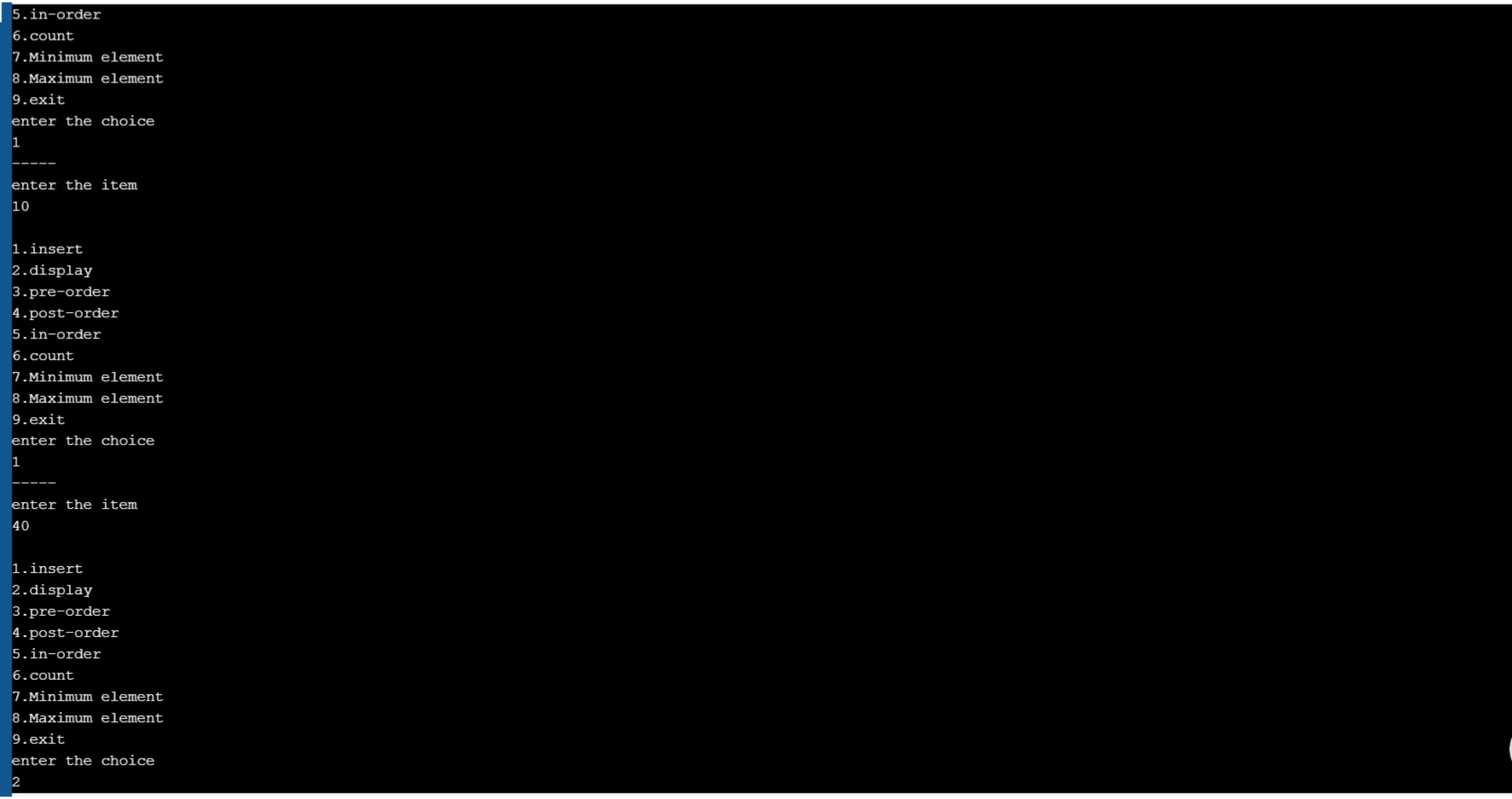
```
        }
}
void main()
{
    int item,choice,c;
    NODE root=NULL;
    for(;;)
    {
        printf("\n1.insert\n2.display\n3.pre-order\n4.post-order\n5.in-order\n6.count\n7.Minimum element\n8.Maximum element\n9.exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        printf("-----\n");
        switch(choice)
        {
            case 1:printf("enter the item\n");
                 scanf("%d",&item);
                 root=insert(root,item);
                 break;
            case 2:display(root,0);
                 break;
            case 3:preorder(root);
                 break;
            case 4:postorder(root);
                 break;
            case 5:inorder(root);
                break;
            case 6:c=count(root);
                 printf("No. of nodes are: %d\n",c);
                 break;
            case 7:minimum(root);
                 break;
            case 8:maximum(root);
                break;
            default:exit(0);
                 break;

        }
    }
}
```



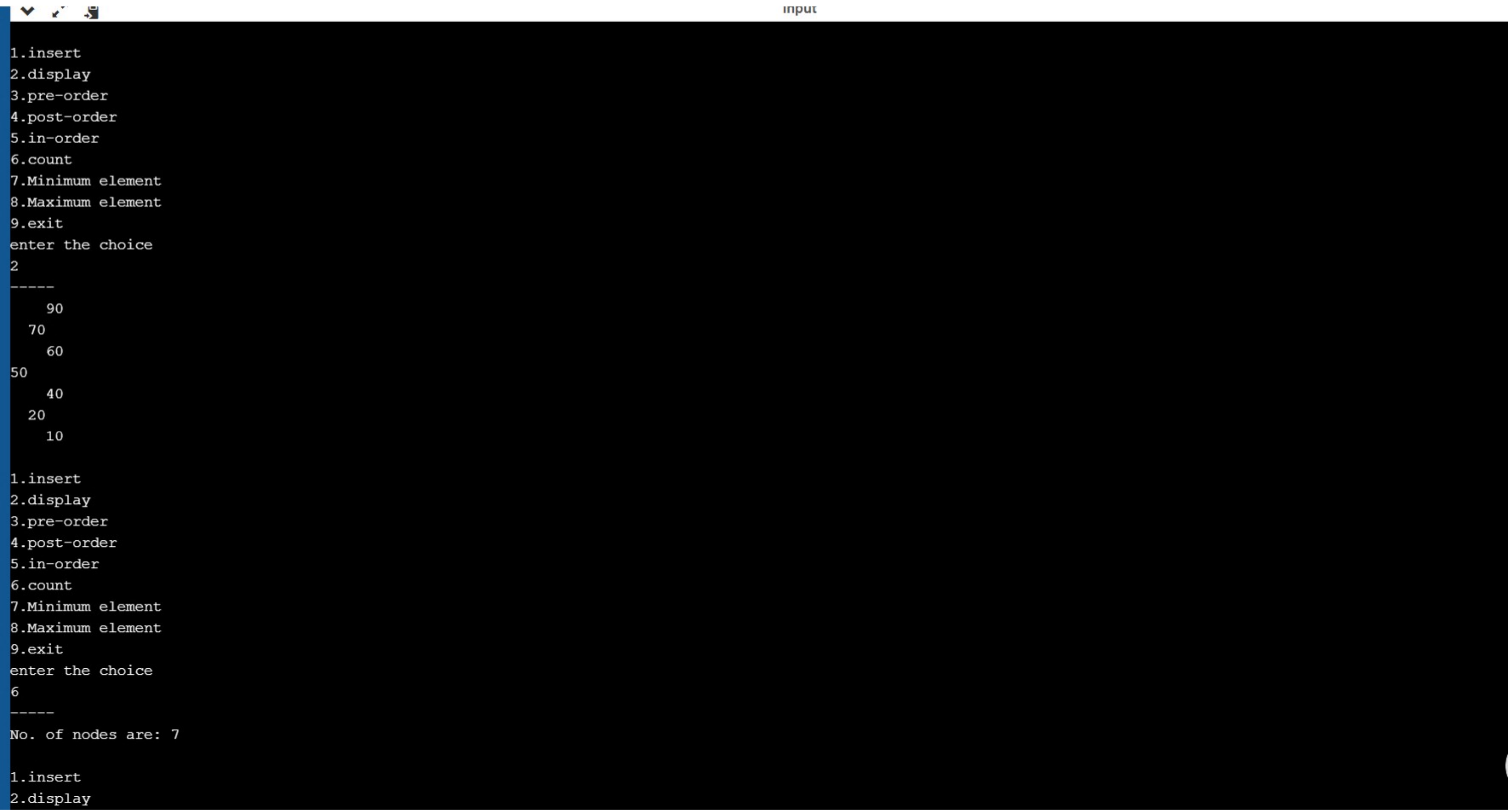
```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
50

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
70

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
```



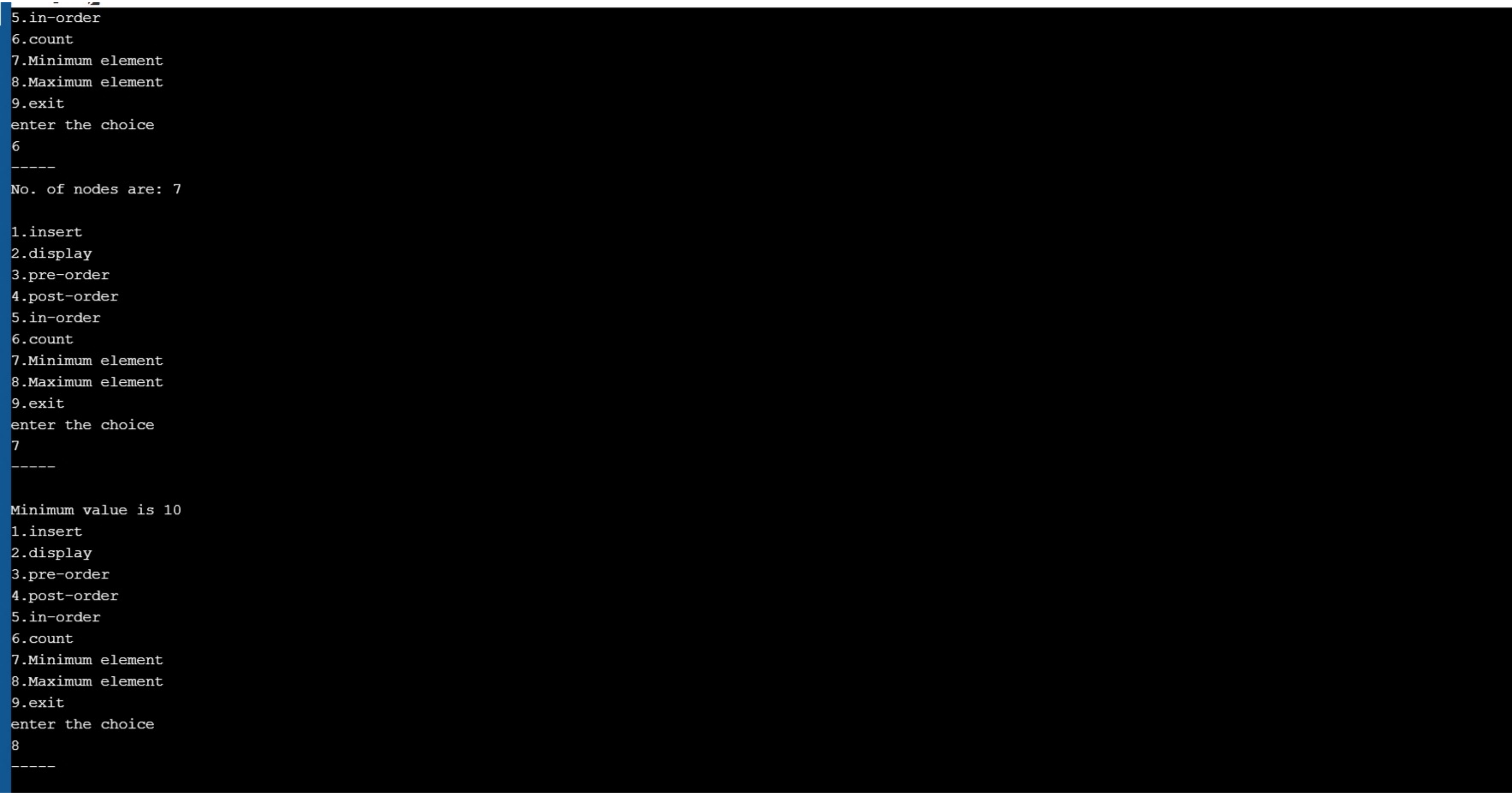
```
enter the choice
1
-----
enter the item
60

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
90

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
20

1.insert
```



```
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
10

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
1
-----
enter the item
40

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
2
```



```
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
2
-----
    90
  70
    60
50
    40
  20
    10

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
6
-----
No. of nodes are: 7

1.insert
2.display
```



```
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
6
-----
No. of nodes are: 7

1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
7
-----


Minimum value is 10
1.insert
2.display
3.pre-order
4.post-order
5.in-order
6.count
7.Minimum element
8.Maximum element
9.exit
enter the choice
8
-----
```