Binary Tree :-

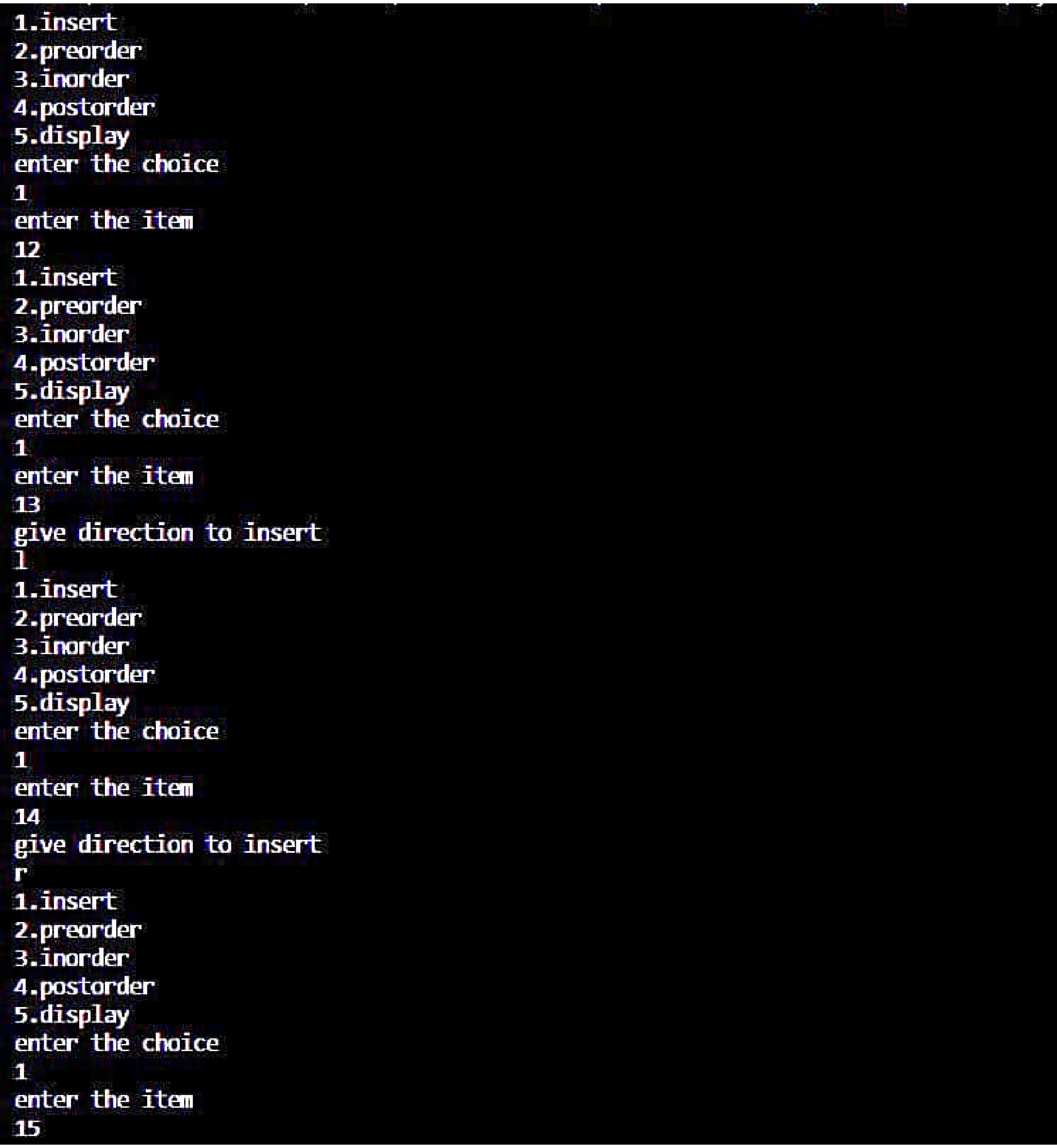
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

struct node
{
    int info;
    struct node *llink;
    struct node *rlink;
}
typedef struct node *NODE;

NODE getnode()
{
    NODE x;
    x=(NODE) mallo(sizeof(struct node));
    if (x==NULL)
    {
        printf("Memory full \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 -> rlink = NULL;
    temp -> llink = NULL;
    if(root == NULL)
        return temp;
    printf(" Give the direction to insert \n");
    scanf(" %d", direction);
    prev = NULL;
    cur
    ~~prev~~ = 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);
```

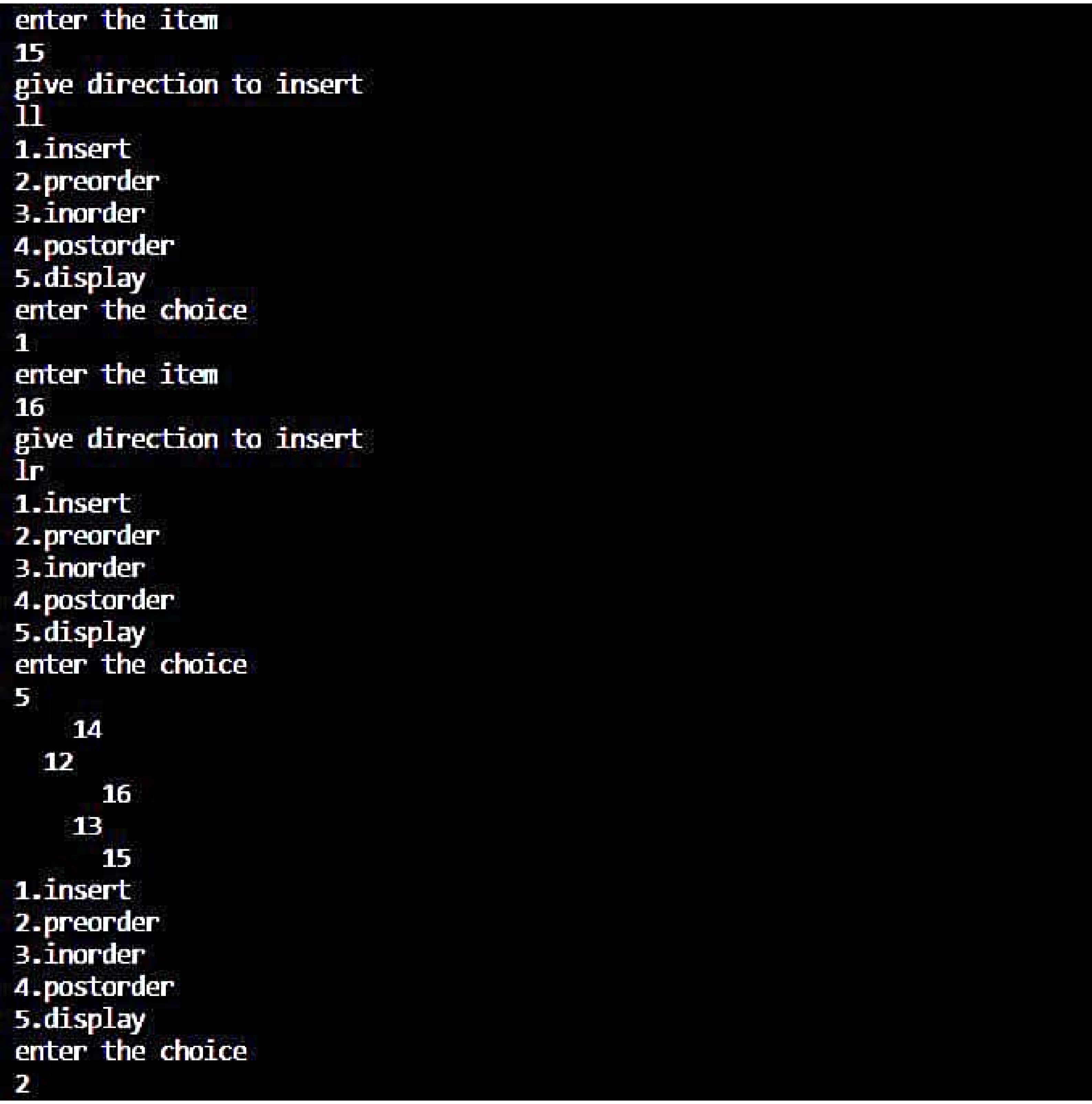
```
void preorder(NODE root)
{
    if(root!=NULL)
    {
        printf("The item is %d\n", root->info);
        preorder(root->llink);
        preorder(root->rlink);
    }
}

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);
    }
}
```

```
~~void create()~~   NODE  insert_bst(NODE root, int item)
{                   {
~~NODE root~~
                        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;
                    }
    NODE  delete (NODE root, int item)
    {
        NODE cur, parent, q, suc;
        if(root == NULL)
        {
            printf("Empty\n");
            return root;
        }
        parent = NULL;
        cur= root;
        ~~[illegible]~~
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;
}
```

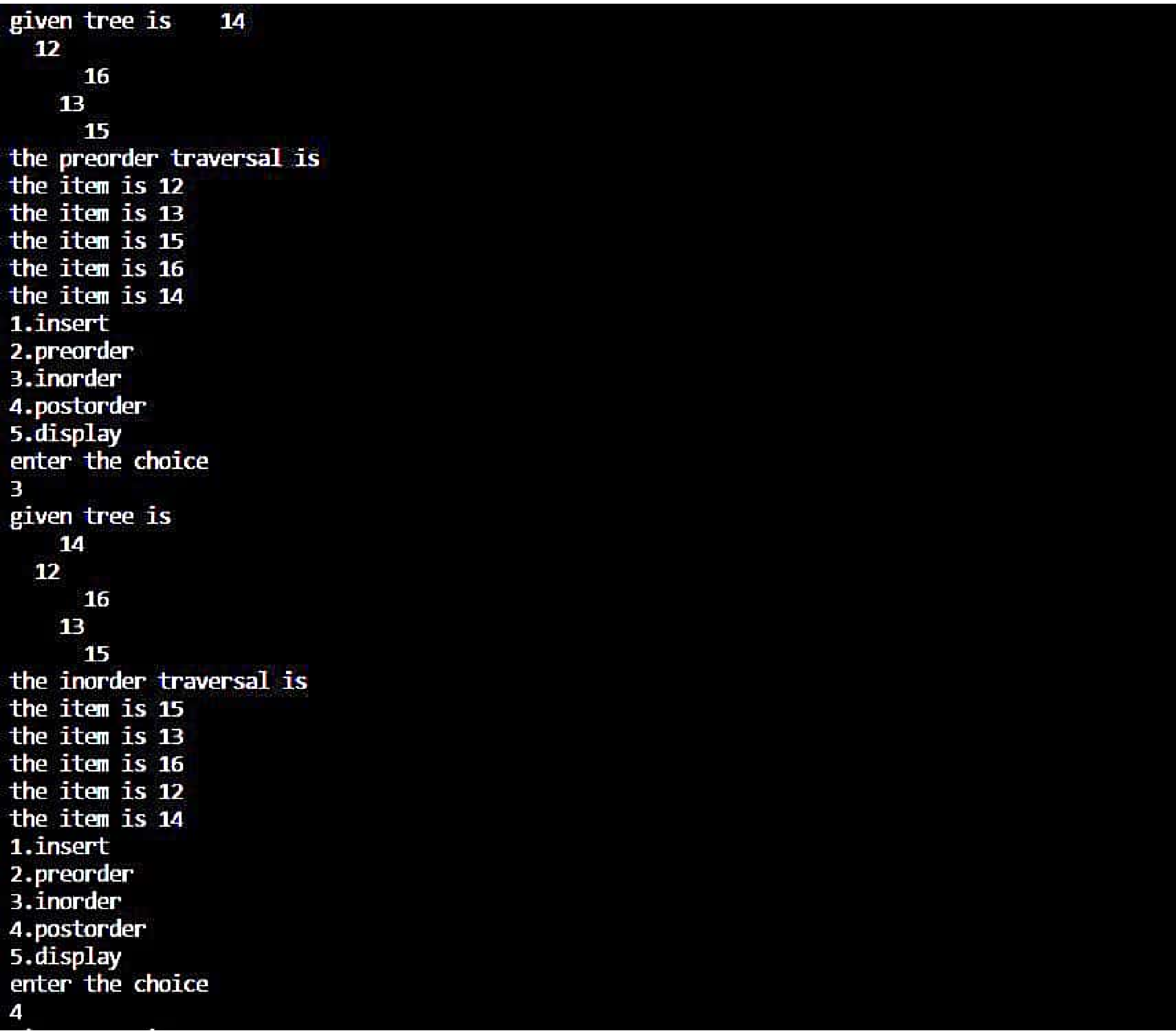
```
void main()
{
  NODE root=NULL;
  int choice,i,item;
  for(;;)
  {
     printf("1. insert \n2. insert in binry tree\n3.Delete Element \n4.
              4. preorder\n5. inorder \n6. Postorder \n7.Display\n");
     printf("Enter the choice\n");
     scanf("%d",&choice);
     switch(choice)
     {
        case 1: printf("Enter the item");
                scanf("%d",&item);
                root = insert(item,root);
                break;
        case 2: printf("Enter the item");
                scanf("%d",&item);
                root= insert_bst(root,item);
                break;
        case 3: printf("Enter the item");
                scanf("%d",&item);
                root = delete(root,item);
                break;
        case 4: if(root==NULL)
                {
                   printf("Tree is empty");
                }
                else
                {
                   printf("Given tree is\n");
                   display(root,1);
                   printf("The preorder traversal is\n");
                   preorder(root);
                }
case 5 : 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 6 : if (root ==NULL)
          {
              printf(" Tree is empty \n");
          }
          else
          {
              printf(" Given tree \n");
              display(root,1);
              printf(" the postorder traversal is \n");
              postorder (root);
          }
          break;
case 7 : display( root,1);
          break;
default : exit (0);
}
}
}
```



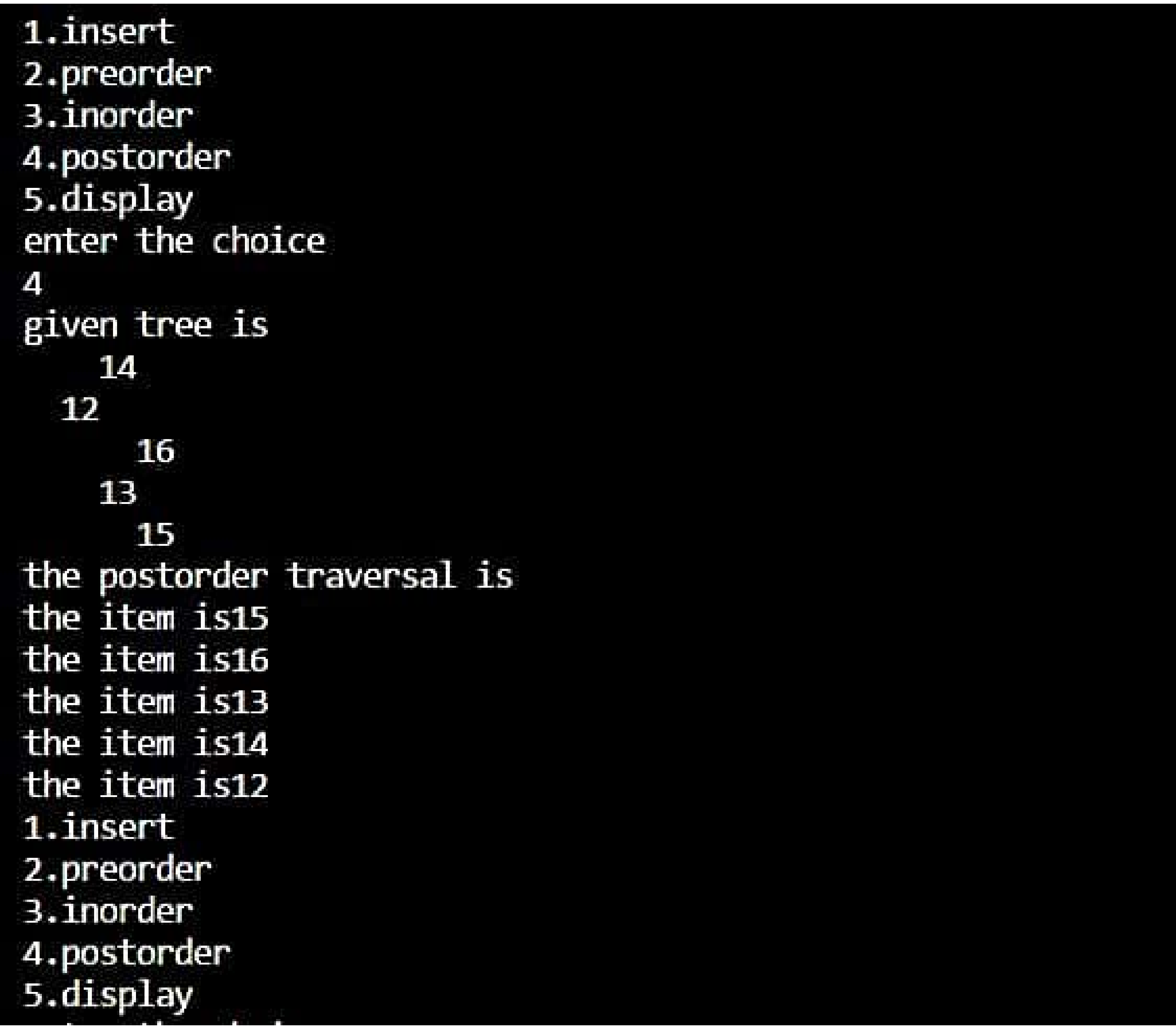
```
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
12
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
13
give direction to insert
l
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
14
give direction to insert
r
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
15
```



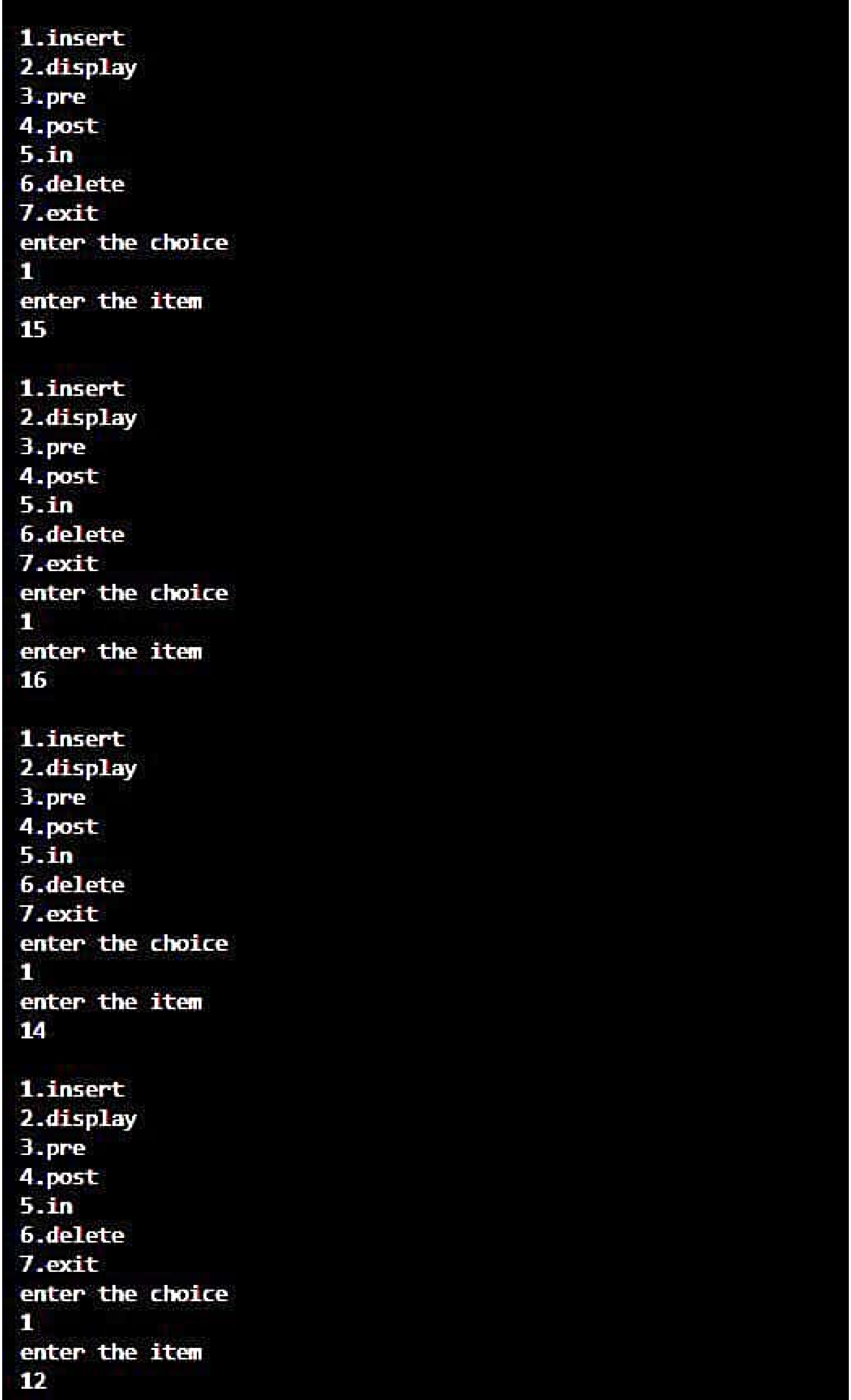
```
enter the item
15
give direction to insert
ll
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
16
give direction to insert
lr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    14
  12
      16
    13
      15
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
2
```



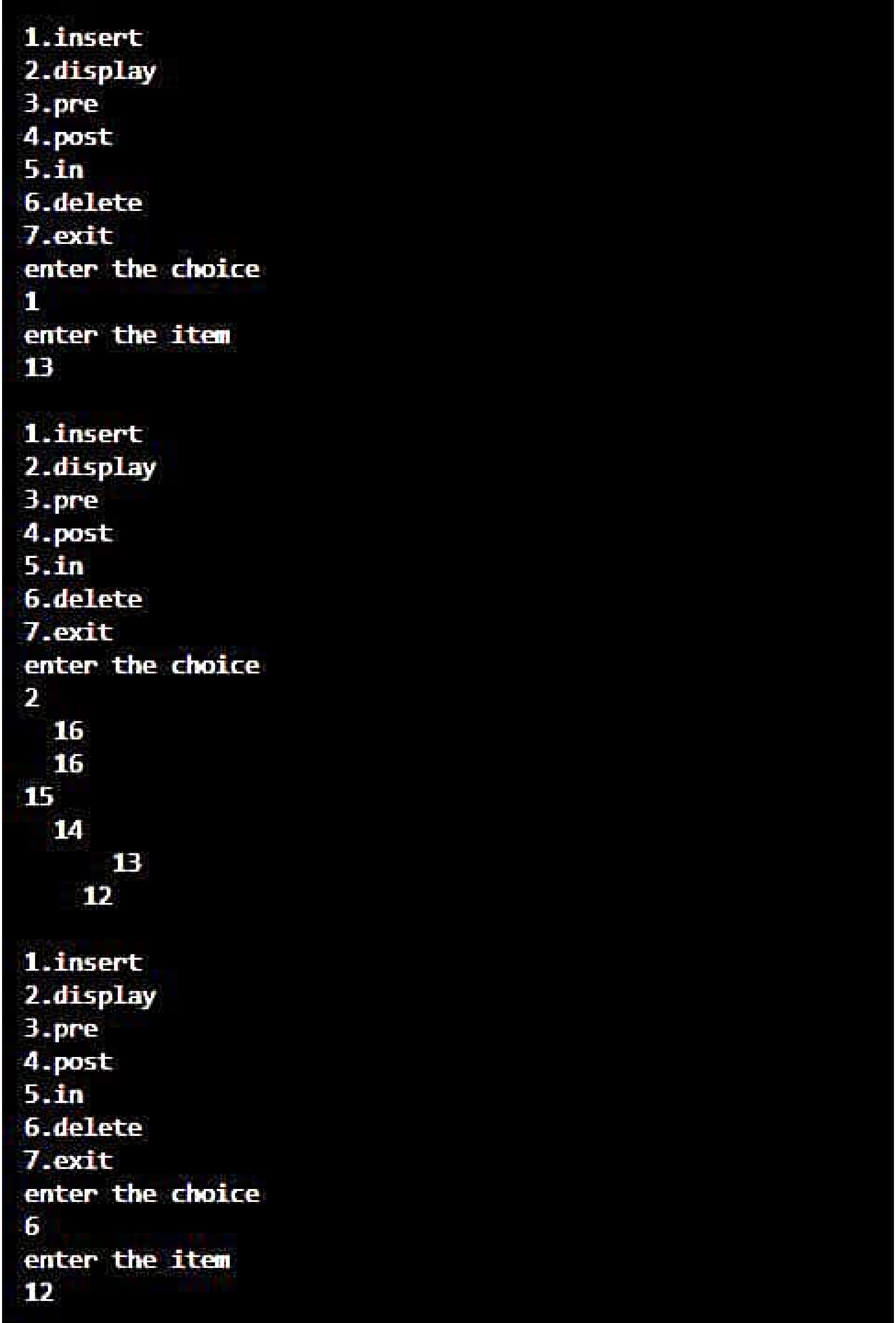
```
given tree is    14
  12
      16
    13
      15
the preorder traversal is
the item is 12
the item is 13
the item is 15
the item is 16
the item is 14
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
given tree is
    14
  12
      16
    13
      15
the inorder traversal is
the item is 15
the item is 13
the item is 16
the item is 12
the item is 14
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
```



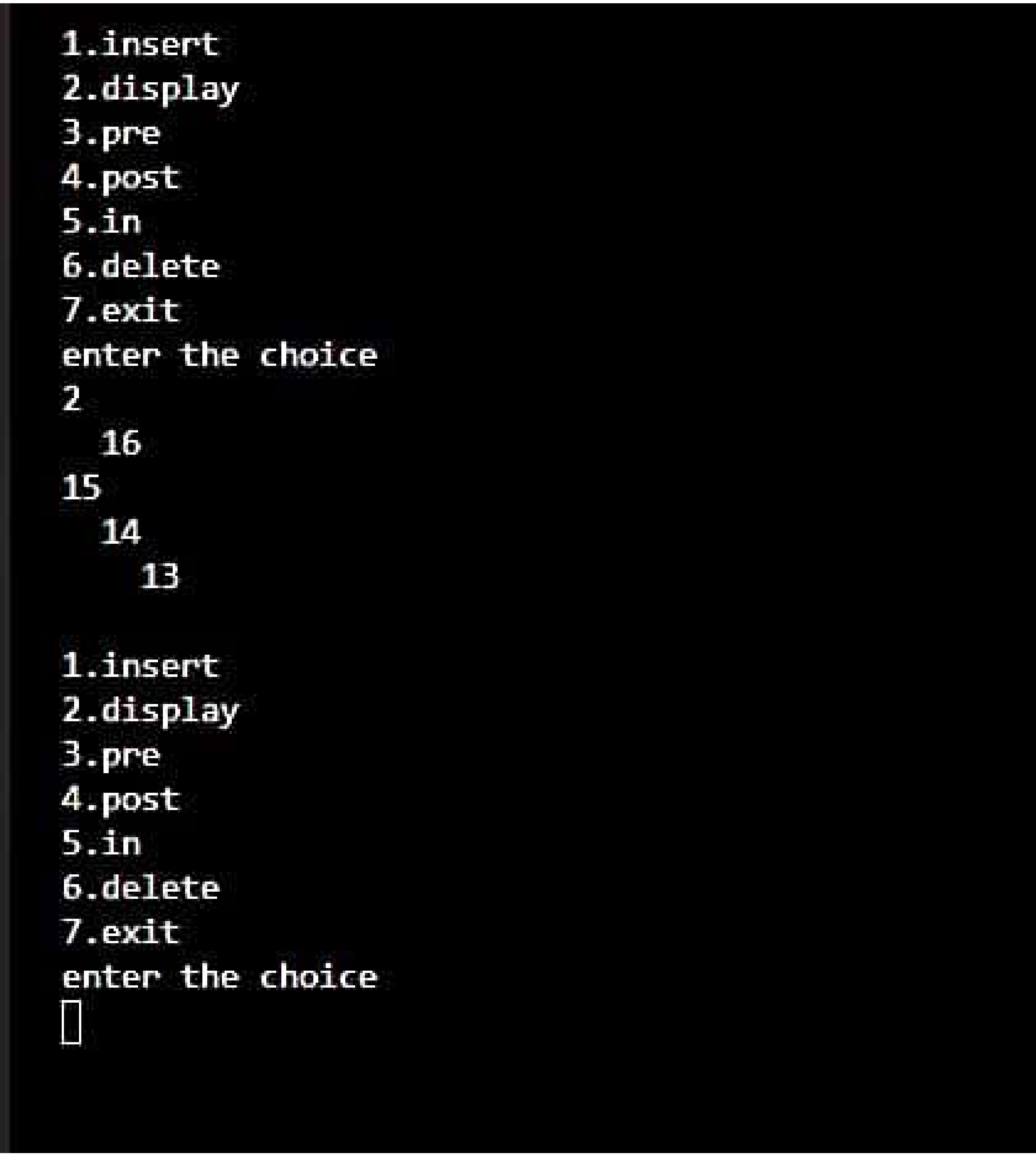
```
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
given tree is
    14
  12
      16
    13
      15
the postorder traversal is
the item is15
the item is16
the item is13
the item is14
the item is12
1.insert
2.preorder
3.inorder
4.postorder
5.display
```



```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
15

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
16

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
14

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
12
```



```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
13

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
  16
  16
15
  14
      13
    12

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
12
```


```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
  16
15
  14
    13

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
```