→ DS Lab Program 10 →

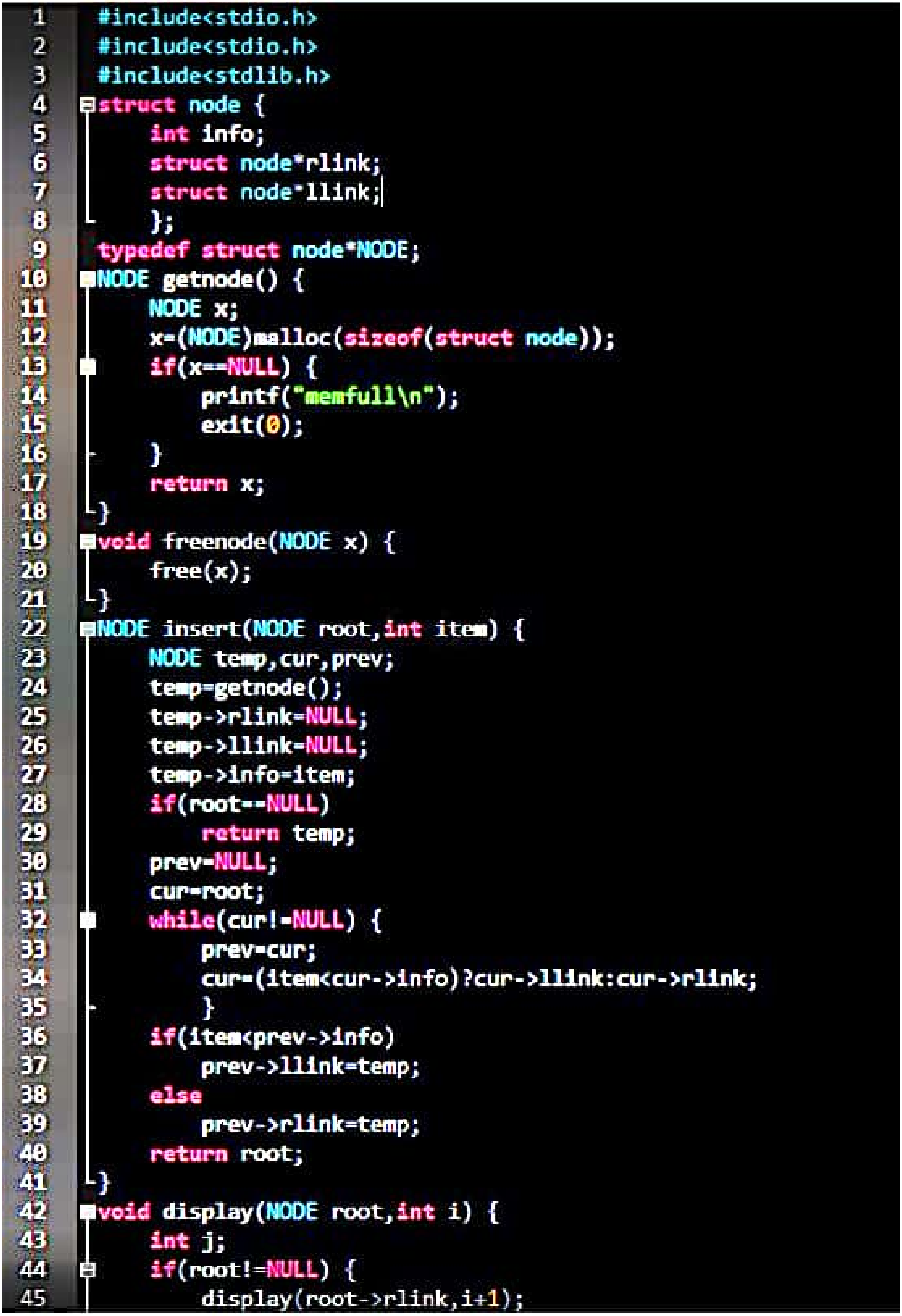
```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int info;
  struct Node *rlink;
  struct Node *llink;
};
typedef struct Node node;
node getnode ();
  node x
  x=(node) malloc (sizeof (node));
  if (x==NULL) {
      printf ("Memory full \n");
      exit (0);
  }
  return x;
}
void freenode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

Date / /

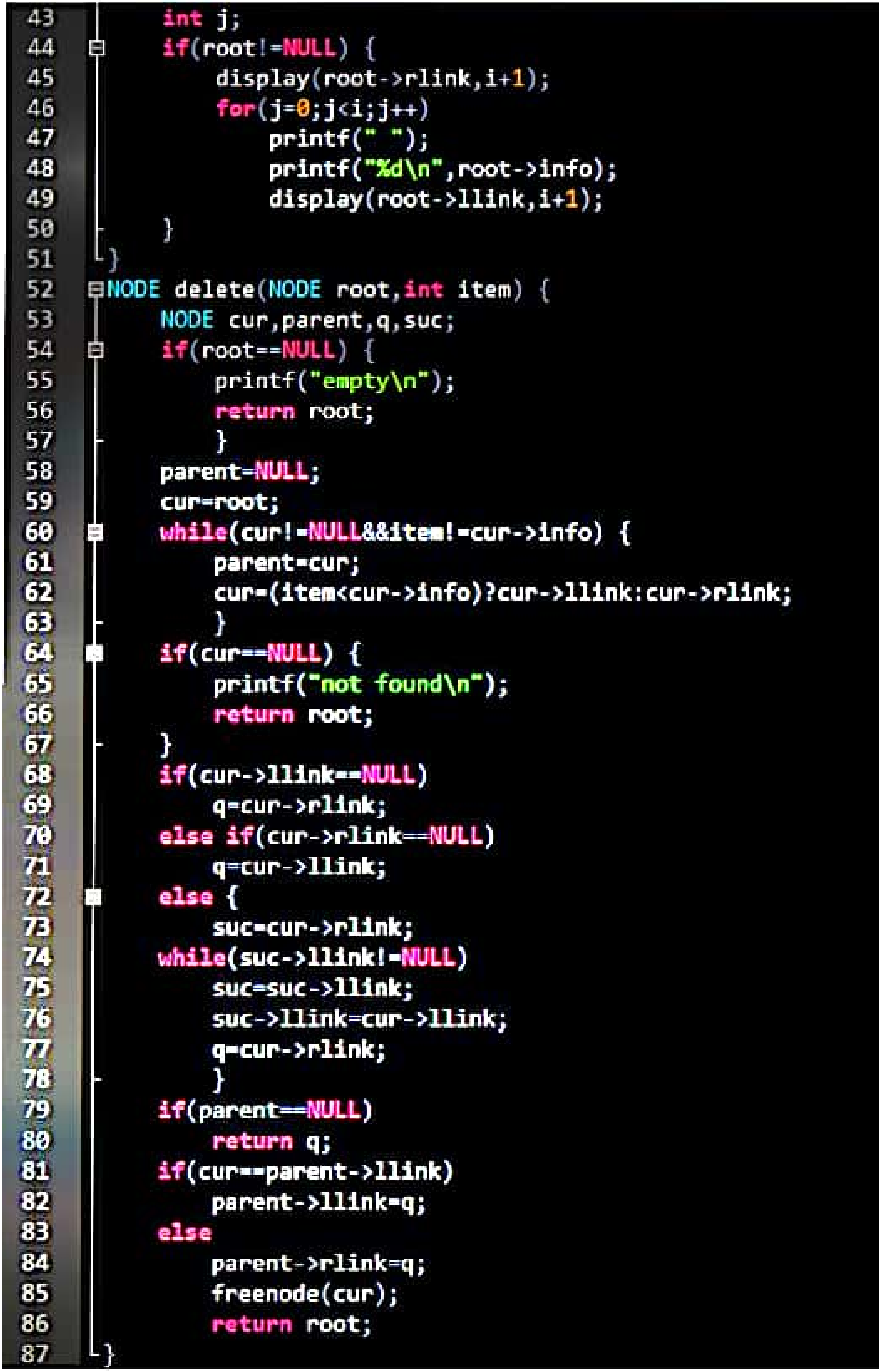
```
node insert (node root, int item) {
  node temp, prev, cur;
  temp = getnode ();
  temp -> rlink = NULL;
  temp -> llink = NULL;
  temp -> info = item;
  if (root == NULL)
       return temp;
  prev = NULL;
  cur = root;
  while (cur != NULL) {
       prev = cur;
       cur = (item < cur -> info)? 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~~ 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);
     printf ( "%d\n", root -> info);
   }
}
void postorder (node root) {
  if (root != NULL) {
     postorder (root -> llink);
     postorder (root -> rlink);
     ~~pos~~ printf ("%d \n", root -> info);
   }
}
void inorder (node root) {
   if (root != NULL) {
      inorder (root -> llink);
      printf ("%d\n", root -> info);
      inorder (root-> rlink);
    }
}
int main () {
  int item, choice;
  node root = NULL;
  for(;;) {
   printf ("\n1. Insert \n2. Display \n3.
      Pre \n4. Post \n5 In \n6. ~~Deltete~~ Delete \n
       7. Exit \n");
   printf ("Enter the choice\n");
   scanf ("%d", &choice);
   switch (choice) {
```

```
case 1: printf ("Enter 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: printf ("Enter the item \n");
        scanf ("%d", &item);
        root= delete (root, item);
        break;
default: exit (0);
         break;
   }
  }
}.
```

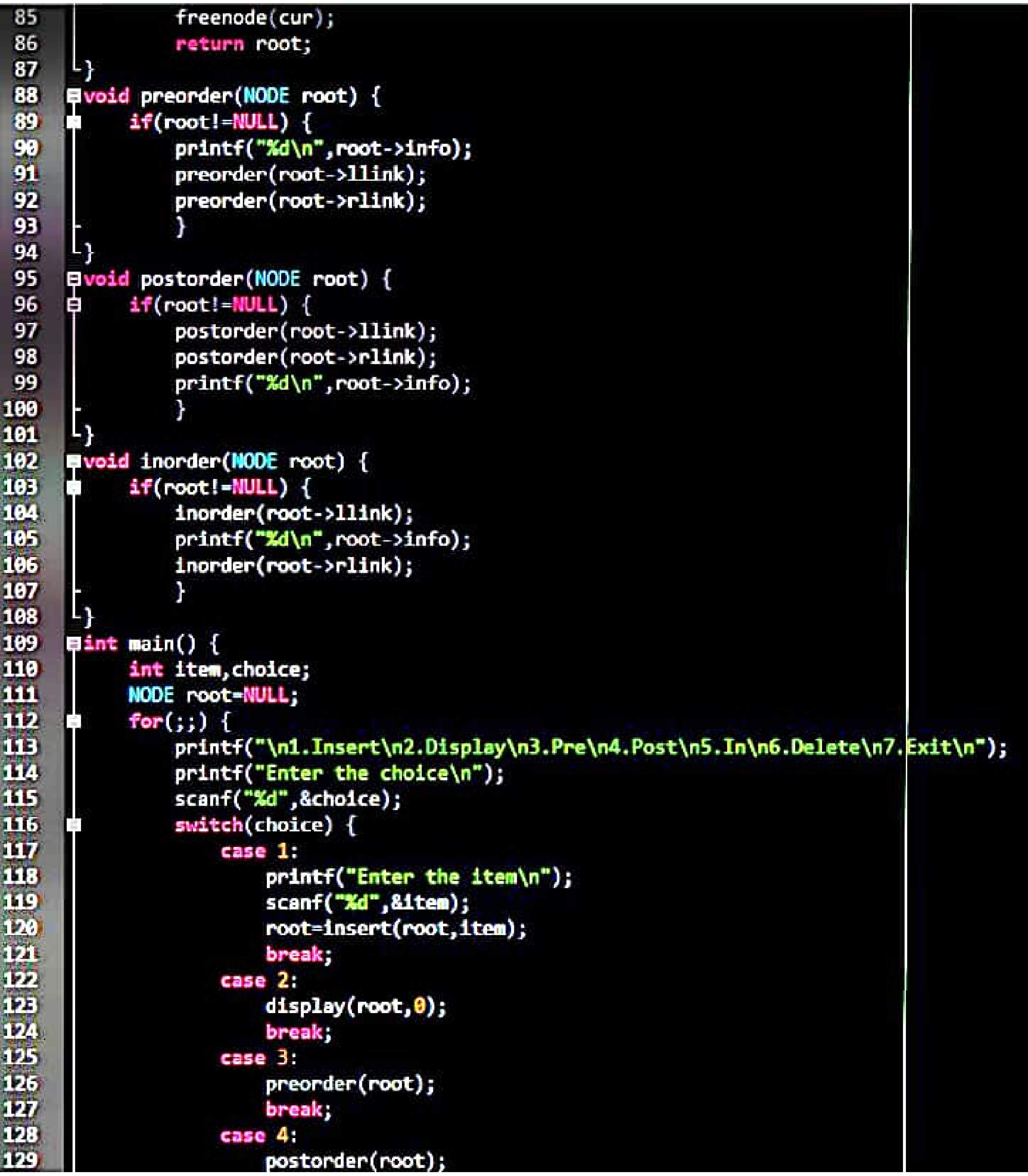


```c
#include<stdio.h>
#include<stdio.h>
#include<stdlib.h>
struct node {
    int info;
    struct node*rlink;
    struct node*llink;
    };
typedef struct node*NODE;
NODE getnode() {
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL) {
        printf("memfull\n");
        exit(0);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

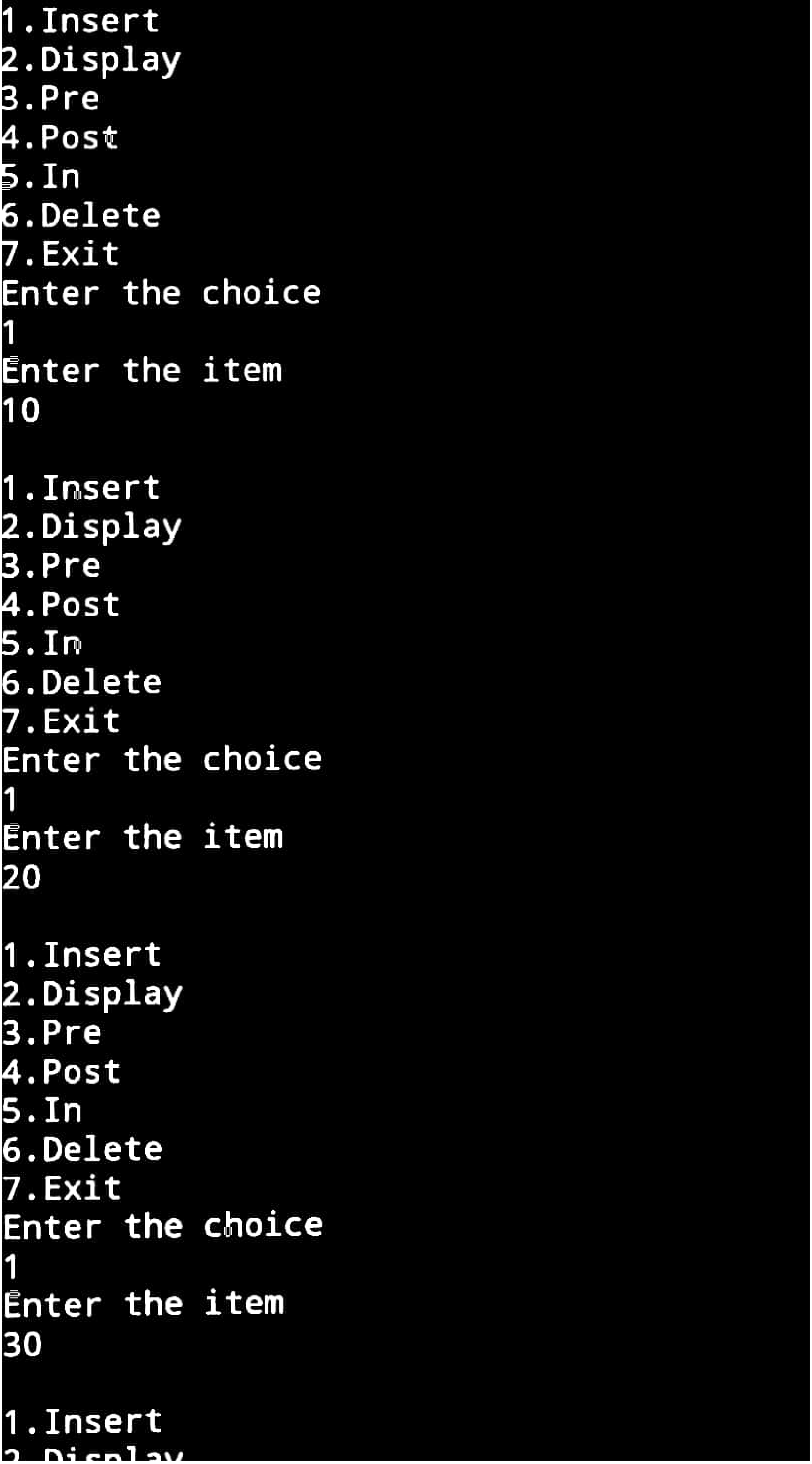


```c
    int j;
    if(root!=NULL) {
        display(root->rlink,i+1);
        for(j=0;j<i;j++)
            printf(" ");
            printf("%d\n",root->info);
            display(root->llink,i+1);
    }
}
NODE delete(NODE root,int item) {
    NODE cur,parent,q,suc;
    if(root==NULL) {
        printf("empty\n");
        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;
}
```

Scanned with CamScanner



```c
        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);
        }
}
int main() {
    int item,choice;
    NODE root=NULL;
    for(;;) {
        printf("\n1.Insert\n2.Display\n3.Pre\n4.Post\n5.In\n6.Delete\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(root,item);
                break;
            case 2:
                display(root,0);
                break;
            case 3:
                preorder(root);
                break;
            case 4:
                postorder(root);
```

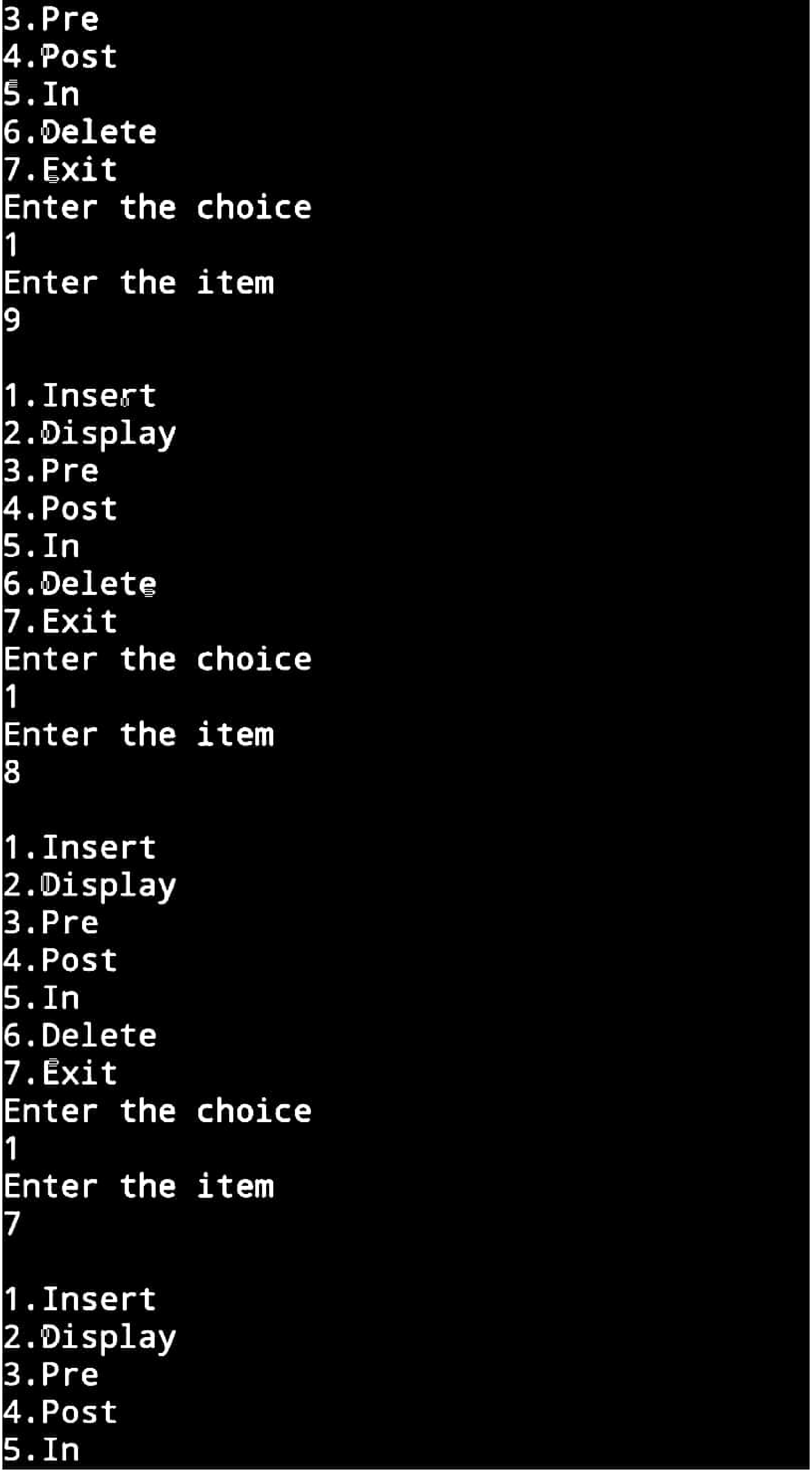


```
1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
1
Enter the item
10

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
1
Enter the item
20

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
1
Enter the item
30

1.Insert
2.Display
```

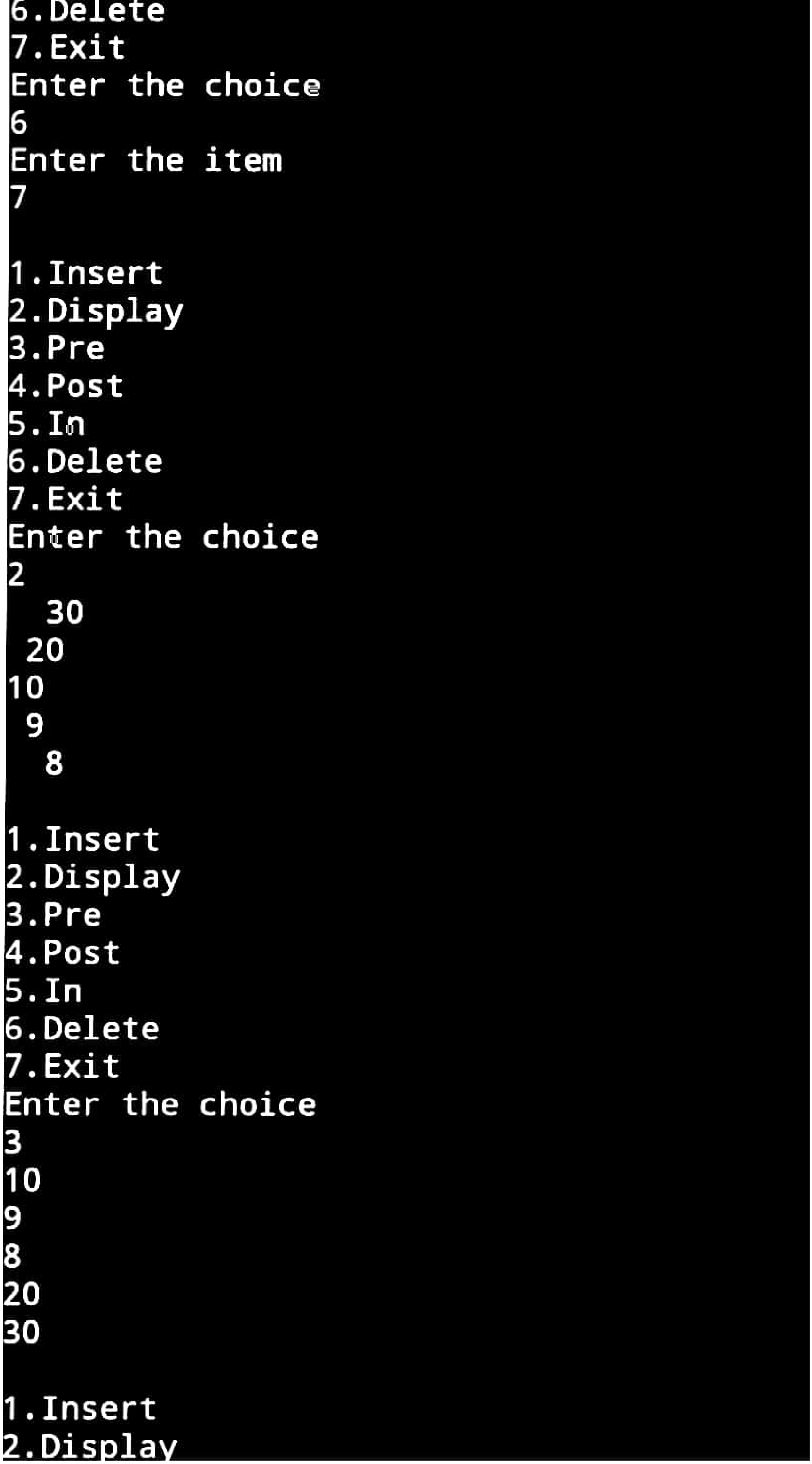


```
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
1
Enter the item
9

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
1
Enter the item
8

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
1
Enter the item
7

1.Insert
2.Display
3.Pre
4.Post
5.In
```

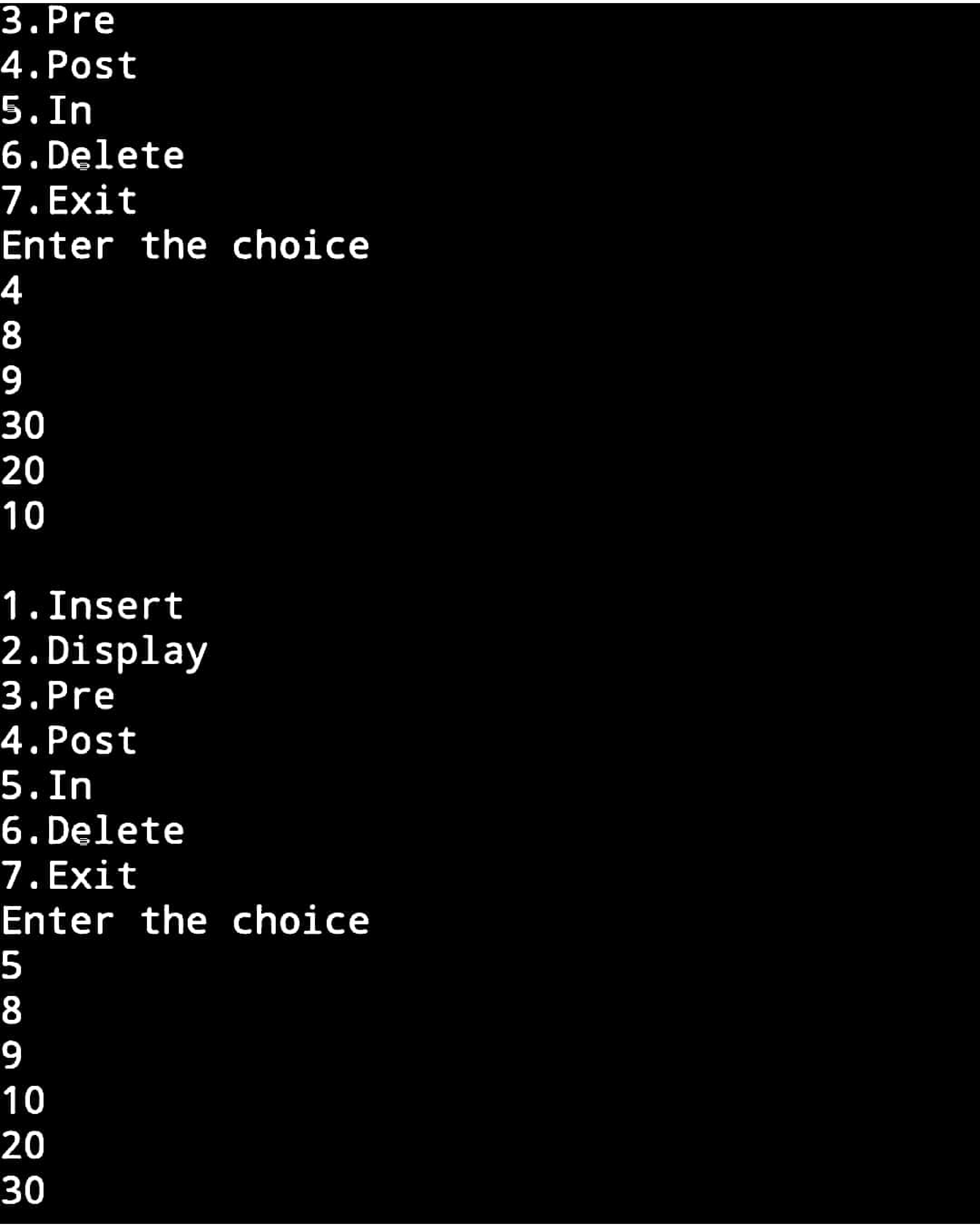


```
6.Delete
7.Exit
Enter the choice
6
Enter the item
7

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
2
  30
 20
10
 9
  8

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
3
10
9
8
20
30

1.Insert
2.Display
```



```
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
4
8
9
30
20
10

1.Insert
2.Display
3.Pre
4.Post
5.In
6.Delete
7.Exit
Enter the choice
5
8
9
10
20
30
```