Lab-Program-5

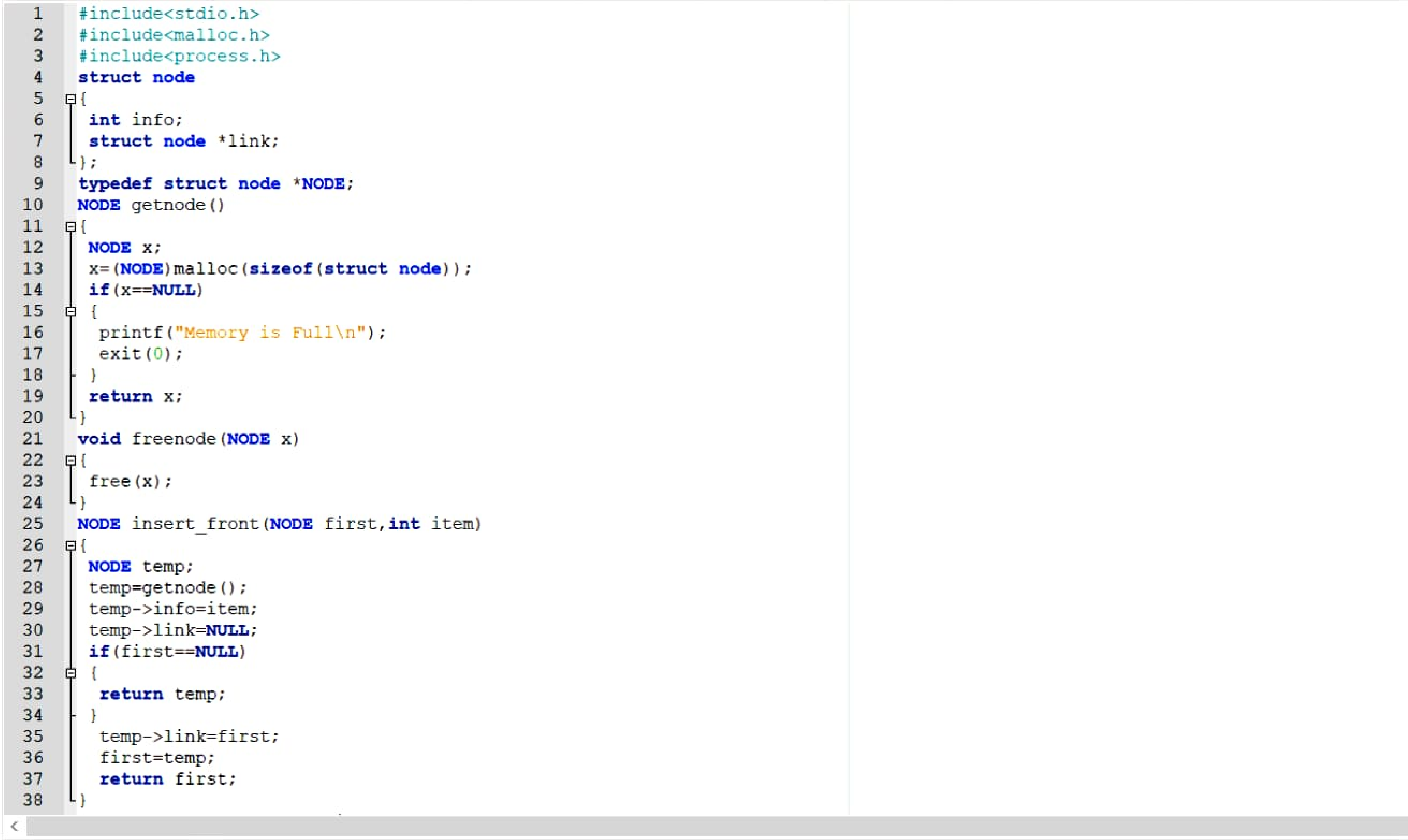
```
#include <stdio.h>
#include <malloc.h>
#include <Process.h>
 struct node
 {
  int info;
  struct node *link;
 };
 typedef struct node *NODE;
 NODE getnode()
 {
  NODE x;
  x=(NODE)malloc(sizeof(struct node));
  if(x==NULL)
  {
   Printf("Memory is ~~F~~ULL\n");
    exit(0);
  }
   return x;
 }
 void freenode(NODE x)
 {
  free(x);
 }
```

```
void freenode (NODE x)
{
 free(x);
}

NODE insert_front(NODE first, int item)
{
  NODE temp;
  temp = getnode();
  temp->info=item;
  temp->link = NULL;
  if (first == NULL)
  {
   return temp;
  }
   temp->link=first;
   first=temp;
   return first;
}
```

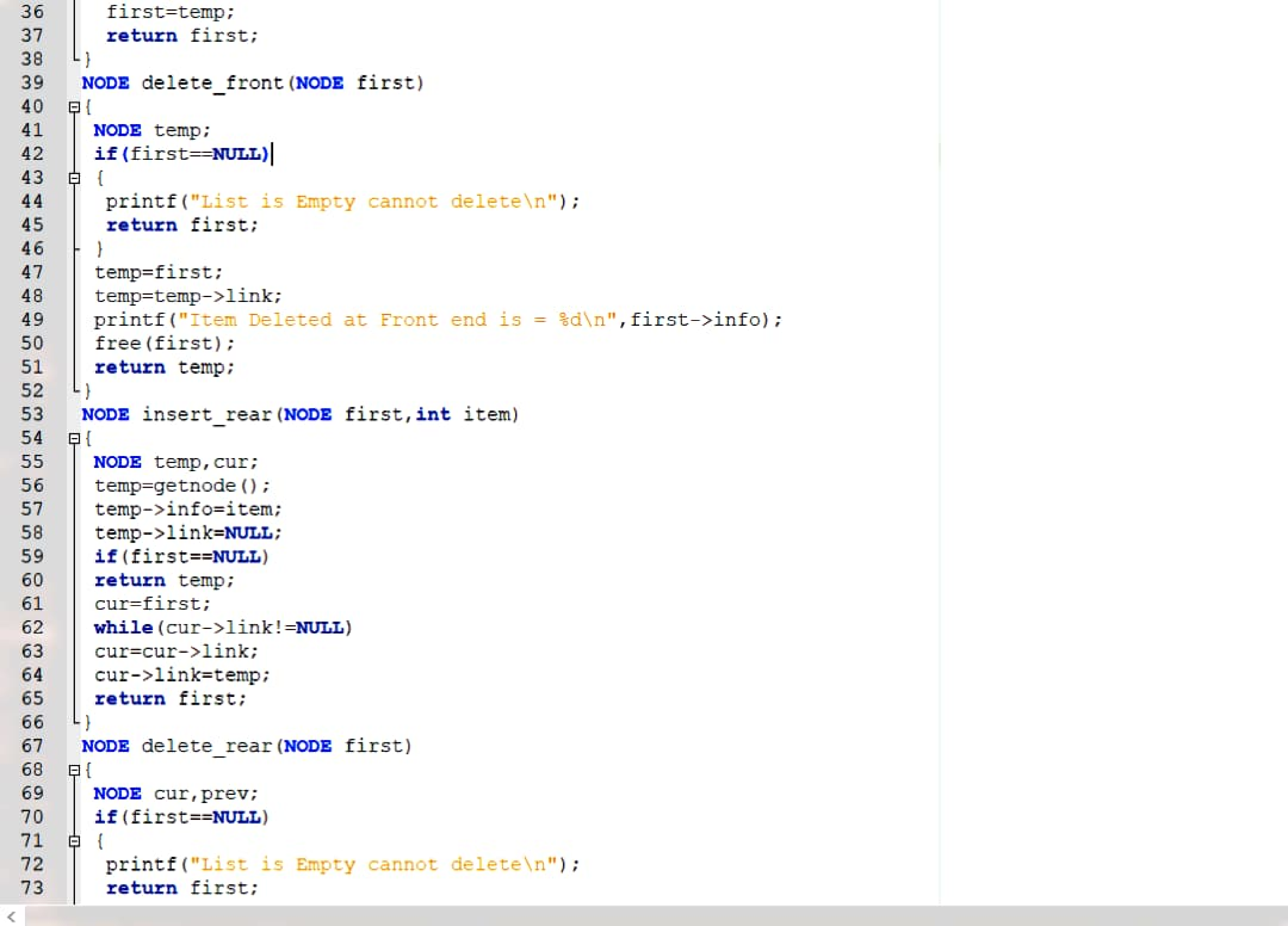
~~temp~~

```
NODE delete_front(NODE first)
{
  NODE temp;
  if (first==NULL)
  {
   Printf("List is Empty cannot delete\n");
   return first;
  }
  temp=first;
  temp=temp->link;
  Printf("Item Deleted at Front end is=%d\n", first
  free(first);
  return temp;
}
NODE insert_rear(NODE first, int item)
{
  NODE temp, cur;
  temp= getnode ();
  temp->info=item;
  temp->link=NULL;
  if (first ==NULL)
   return temp;
  cur= first;
  while (cur->link != NULL)
  cur = cur->link;
  cur->link = temp;
  return first;
}
NODE delete_rear(NODE first)
{
  NODE cur, Prev;
  if (first == NULL)
  {
    Printf("List is empty cannot delete\n");
    return first;
  }
  if (first->link == NULL)
  {
    Printf("Item Deleted is %d\n", first->info);
    free (first);
    return NULL;
  }
  Prev = NULL;
  cur = first;
  while (cur->link != NULL)
  {
    Prev = cur;
    cur = cur->link;
  }
```

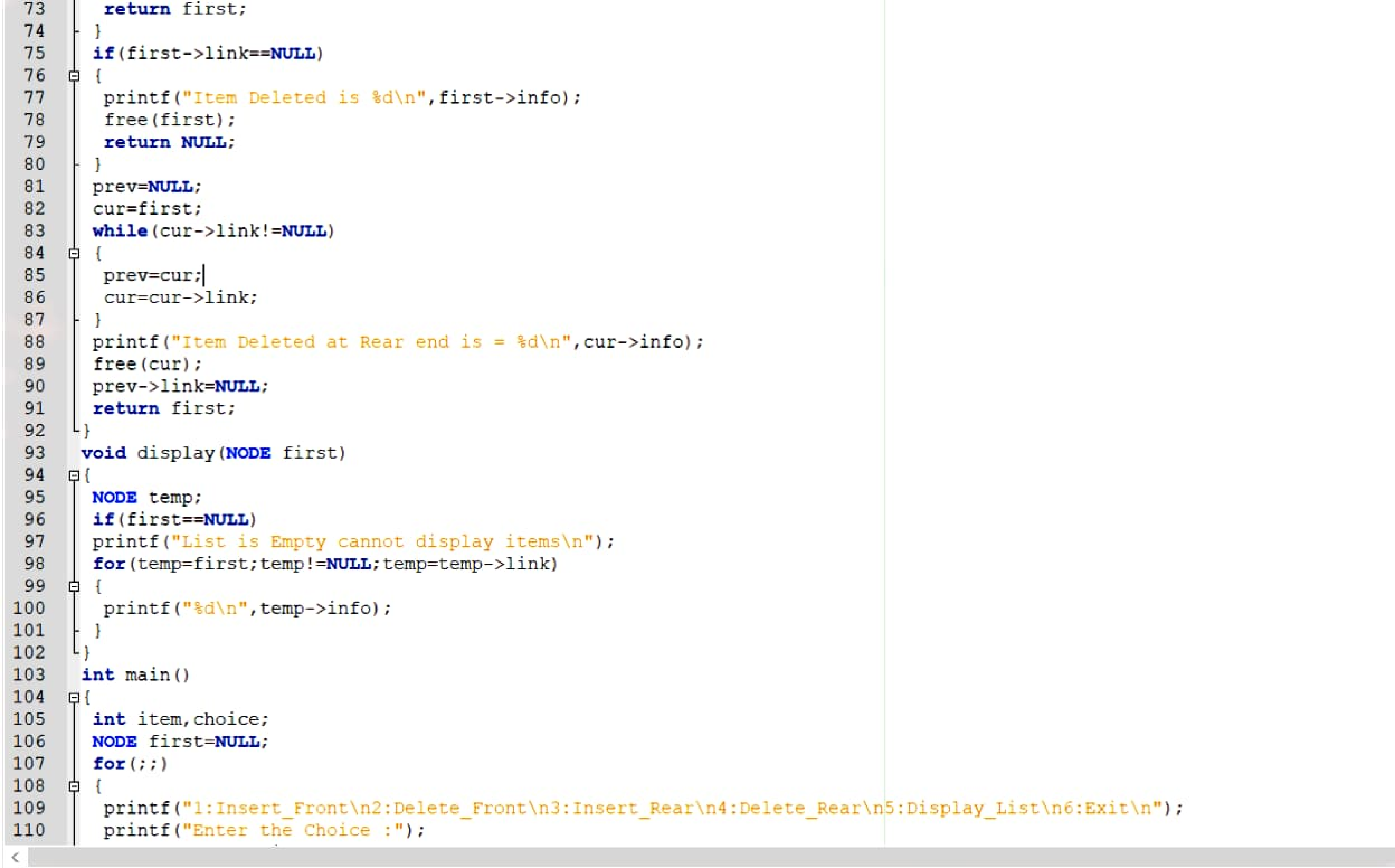
```
Printf ("Item Deleted at Rear and is
        = %d\n", cur->info);
free(cur);
 Prev->link = NULL;
 return first;
}
void display (NODE first)
{
  NODE temp;
  if (first == NULL)
  Printf ("List is Empty cannot display items\n");
  for (temp = first; temp! = NULL; temp = temp->link)
  {
   Printf ("%d\n", temp->info);
   }
 }
 int main()
{
  int item, choice;
  NODE first = NULL;
  for(;;)
 {
  Printf ("1: Insert_Front \n 2: Delete_Front \n
          3: Insert_Rear \n 4: Delete_Rear \n
          5: Display_List \n 6: Exit \n");
 scanf ("%d", &choice);
switch (choice)
{
 case 1: Printf("Enter the Item at Front-end\n")
         scanf("%d", &item);
         first=insert_front(first, item);
         break;

 case 2: first=delete_front(first);
         break;
 case 3: Printf("Enter the Item at Rear-
                                  end\n");
         scanf("%d", &item);
         first=insert_rear(first, item);
         break;

 case 4 : first=delete_rear(first);
          break;

  case 5 : display(first);
           break;

  default : exit(0);
            break;
 }
}
 return 0;
}
```



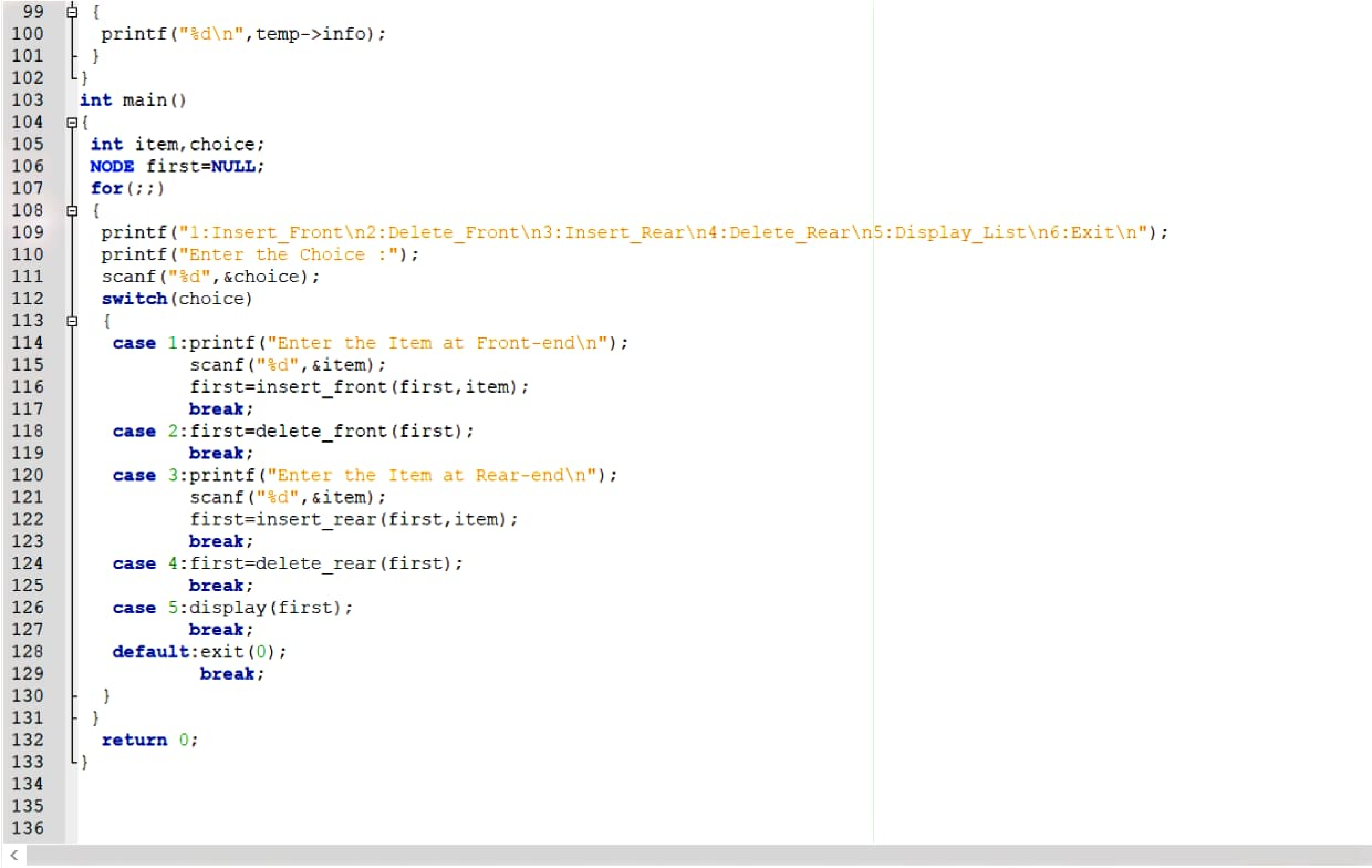
```c
#include<stdio.h>
#include<malloc.h>
#include<process.h>
struct node
{
 int info;
 struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
 NODE x;
 x=(NODE)malloc(sizeof(struct node));
 if(x==NULL)
 {
  printf("Memory is Full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
 free(x);
}
NODE insert_front(NODE first,int item)
{
 NODE temp;
 temp=getnode();
 temp->info=item;
 temp->link=NULL;
 if(first==NULL)
 {
  return temp;
 }
  temp->link=first;
  first=temp;
  return first;
}
```



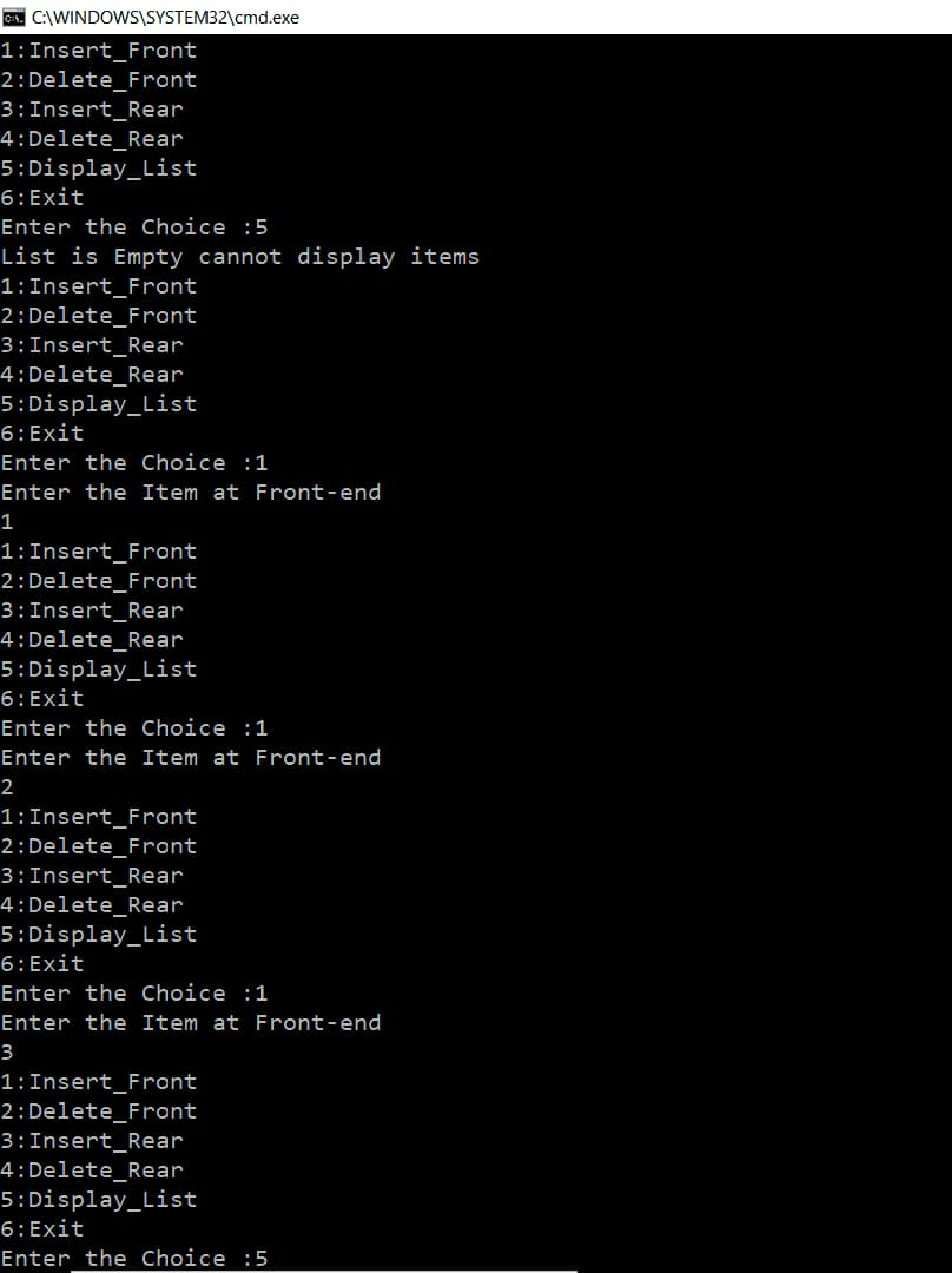
```c
  first=temp;
  return first;
}
NODE delete_front(NODE first)
{
 NODE temp;
 if(first==NULL)
 {
  printf("List is Empty cannot delete\n");
  return first;
 }
 temp=first;
 temp=temp->link;
 printf("Item Deleted at Front end is = %d\n",first->info);
 free(first);
 return temp;
}
NODE insert_rear(NODE first,int item)
{
 NODE temp,cur;
 temp=getnode();
 temp->info=item;
 temp->link=NULL;
 if(first==NULL)
 return temp;
 cur=first;
 while(cur->link!=NULL)
 cur=cur->link;
 cur->link=temp;
 return first;
}
NODE delete_rear(NODE first)
{
 NODE cur,prev;
 if(first==NULL)
 {
  printf("List is Empty cannot delete\n");
  return first;
```



```c
  return first;
 }
 if(first->link==NULL)
 {
  printf("Item Deleted is %d\n",first->info);
  free(first);
  return NULL;
 }
 prev=NULL;
 cur=first;
 while(cur->link!=NULL)
 {
  prev=cur;
  cur=cur->link;
 }
 printf("Item Deleted at Rear end is = %d\n",cur->info);
 free(cur);
 prev->link=NULL;
 return first;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List is Empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
 {
  printf("%d\n",temp->info);
 }
}
int main()
{
 int item,choice;
 NODE first=NULL;
 for(;;)
 {
  printf("1:Insert_Front\n2:Delete_Front\n3:Insert_Rear\n4:Delete_Rear\n5:Display_List\n6:Exit\n");
  printf("Enter the Choice :");
```

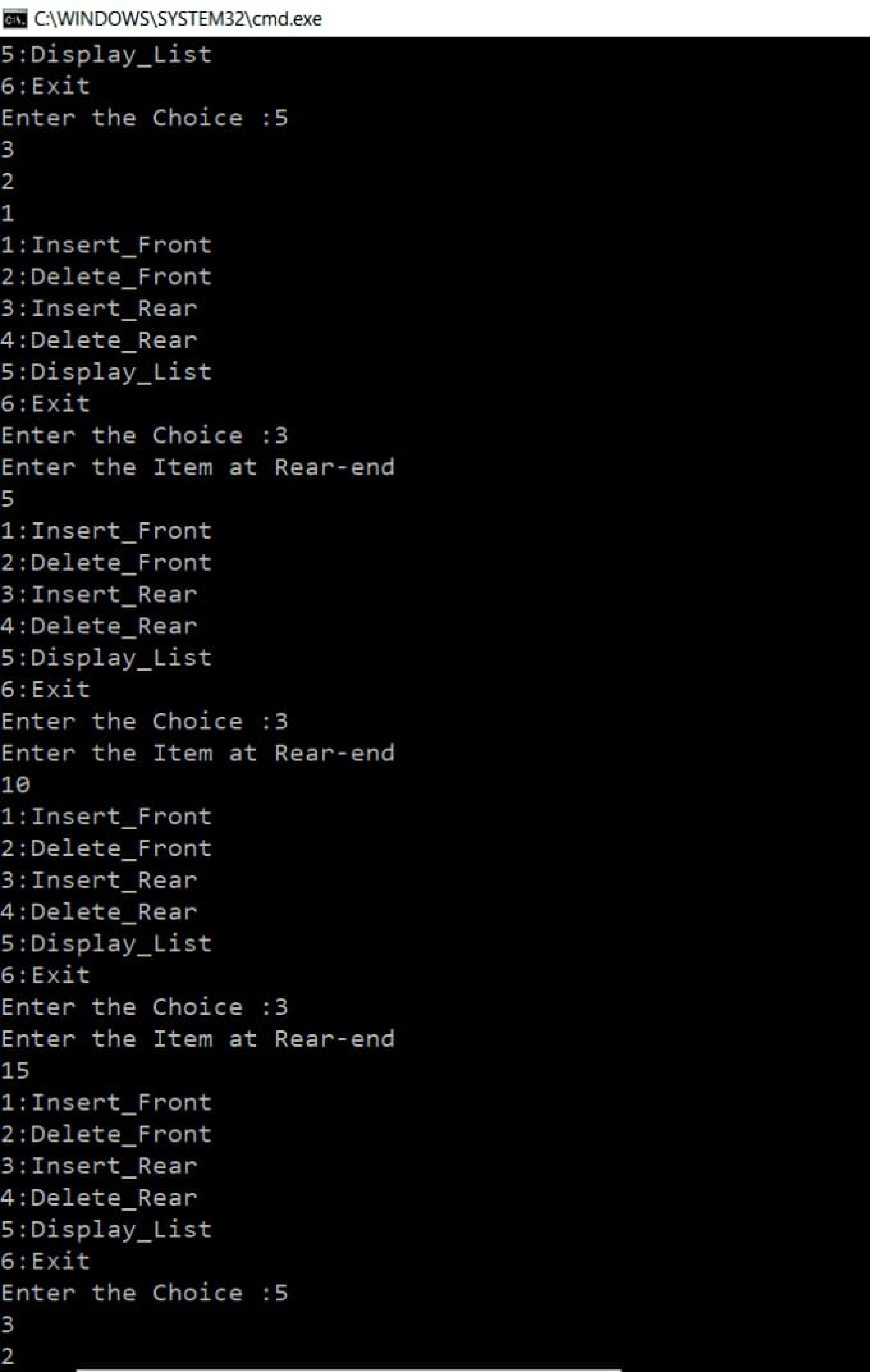


```
  {
   printf("%d\n",temp->info);
  }
}
int main()
{
 int item,choice;
 NODE first=NULL;
 for(;;)
 {
  printf("1:Insert_Front\n2:Delete_Front\n3:Insert_Rear\n4:Delete_Rear\n5:Display_List\n6:Exit\n");
  printf("Enter the Choice :");
  scanf("%d",&choice);
  switch(choice)
  {
   case 1:printf("Enter the Item at Front-end\n");
          scanf("%d",&item);
          first=insert_front(first,item);
          break;
   case 2:first=delete_front(first);
          break;
   case 3:printf("Enter the Item at Rear-end\n");
          scanf("%d",&item);
          first=insert_rear(first,item);
          break;
   case 4:first=delete_rear(first);
          break;
   case 5:display(first);
          break;
   default:exit(0);
           break;
  }
 }
  return 0;
}
```

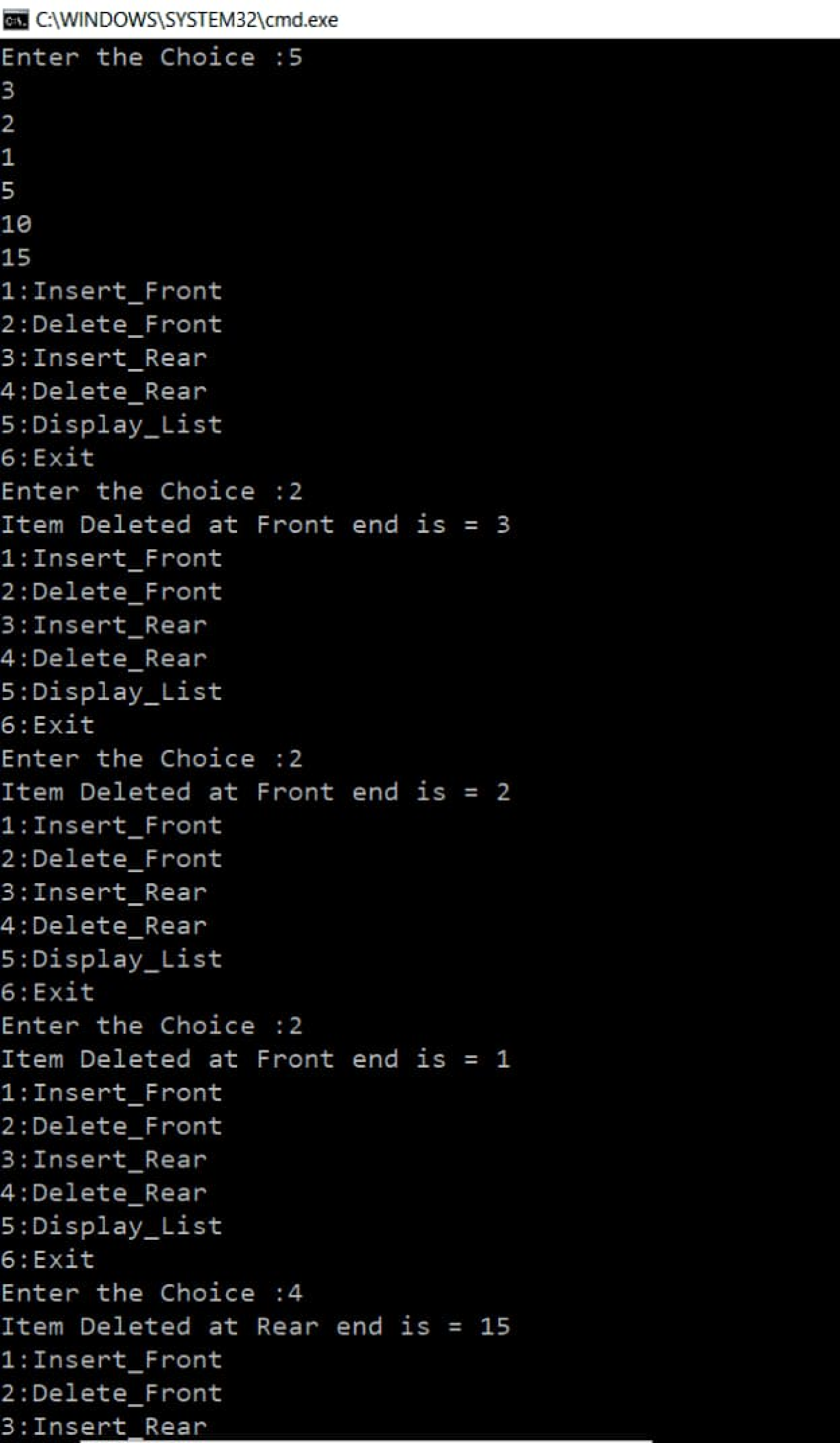


C:\WINDOWS\SYSTEM32\cmd.exe

```
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
List is Empty cannot display items
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :1
Enter the Item at Front-end
1
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :1
Enter the Item at Front-end
2
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :1
Enter the Item at Front-end
3
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
```

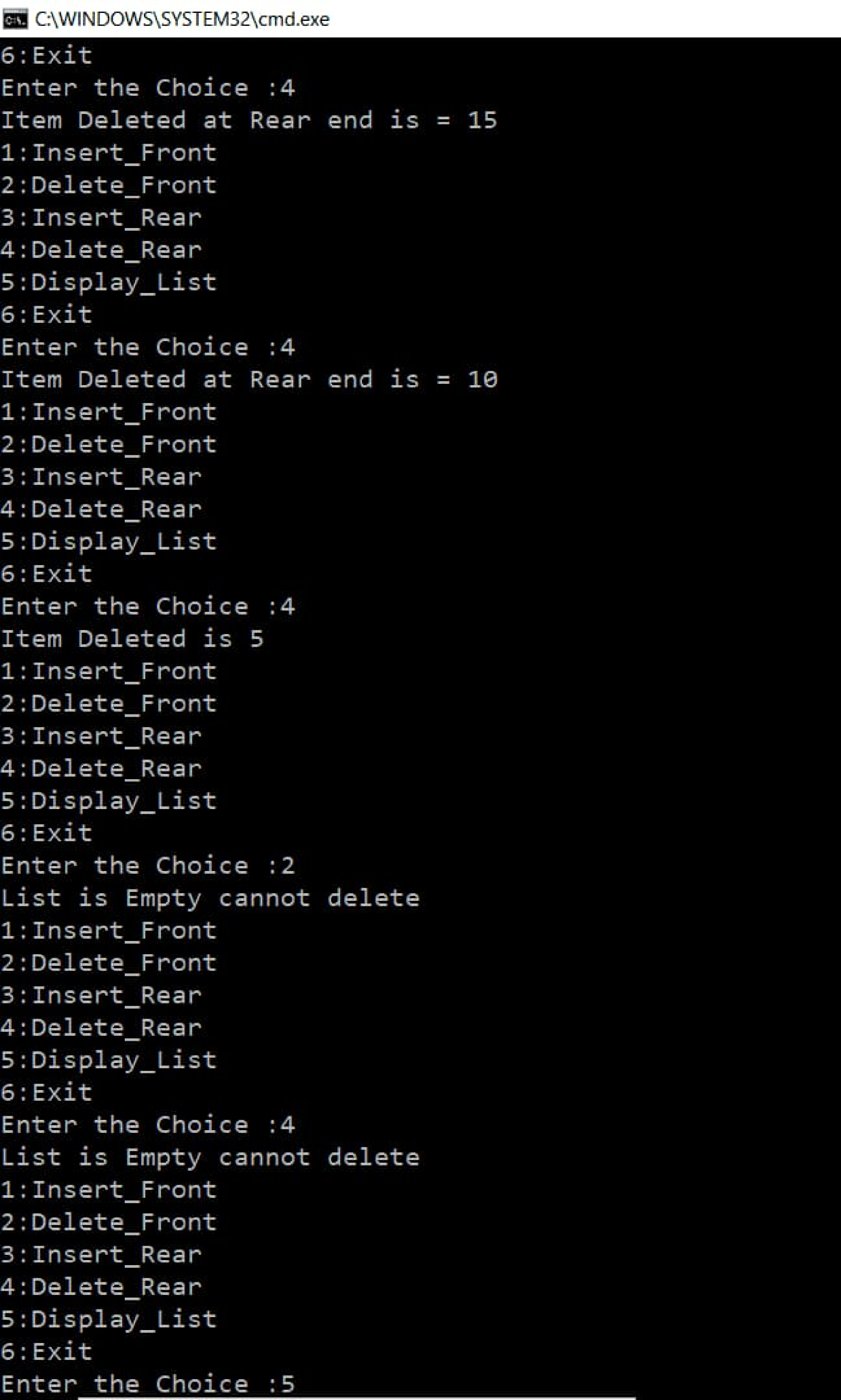


```
C:\WINDOWS\SYSTEM32\cmd.exe
5:Display_List
6:Exit
Enter the Choice :5
3
2
1
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :3
Enter the Item at Rear-end
5
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :3
Enter the Item at Rear-end
10
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :3
Enter the Item at Rear-end
15
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
3
2
```

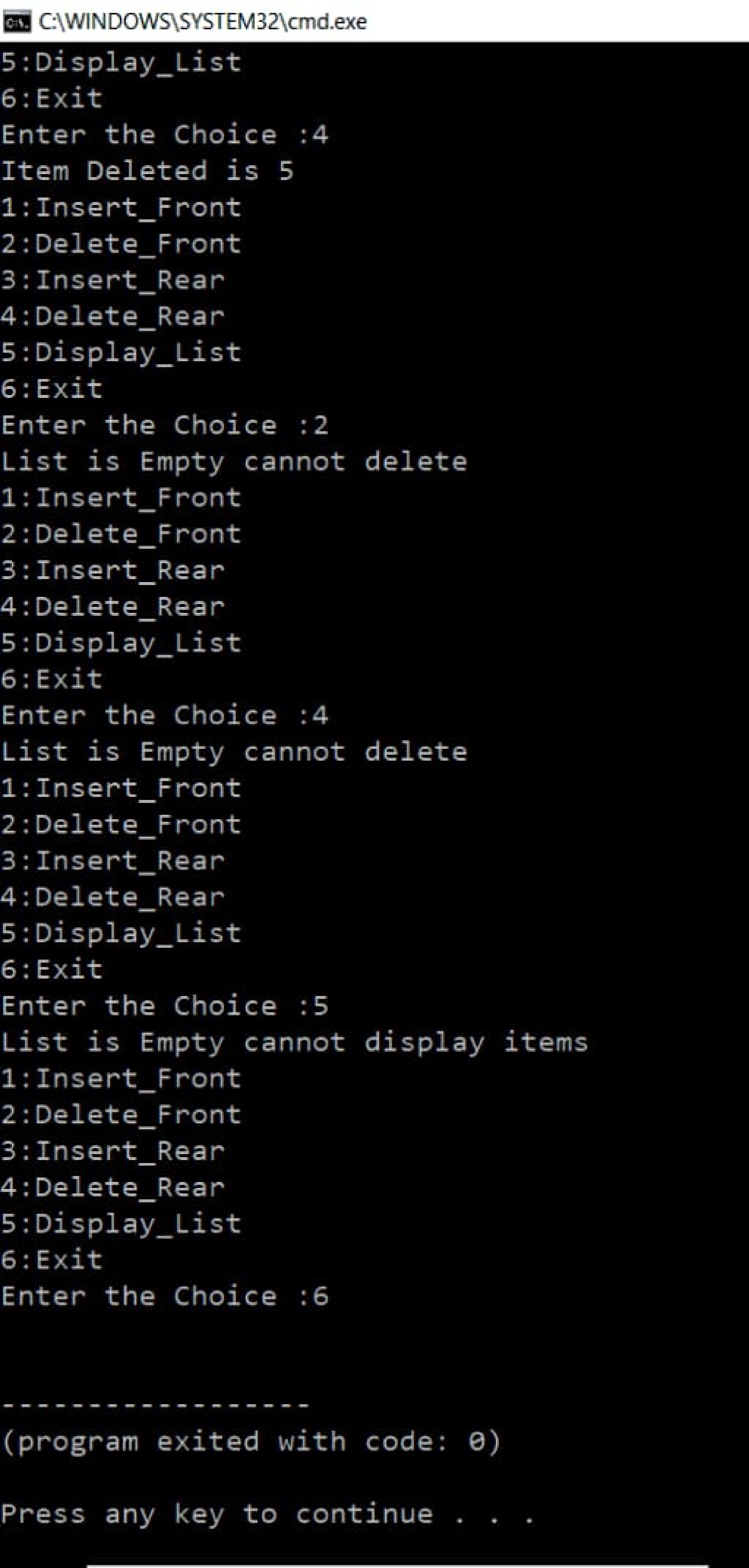


C:\WINDOWS\SYSTEM32\cmd.exe

```
Enter the Choice :5
3
2
1
5
10
15
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :2
Item Deleted at Front end is = 3
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :2
Item Deleted at Front end is = 2
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :2
Item Deleted at Front end is = 1
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
Item Deleted at Rear end is = 15
1:Insert_Front
2:Delete_Front
3:Insert_Rear
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
6:Exit
Enter the Choice :4
Item Deleted at Rear end is = 15
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
Item Deleted at Rear end is = 10
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
Item Deleted is 5
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :2
List is Empty cannot delete
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
List is Empty cannot delete
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
5:Display_List
6:Exit
Enter the Choice :4
Item Deleted is 5
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :2
List is Empty cannot delete
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
List is Empty cannot delete
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
List is Empty cannot display items
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :6



------------------
(program exited with code: 0)

Press any key to continue . . .
```