WAP Implement Single Link List with following operations a) a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

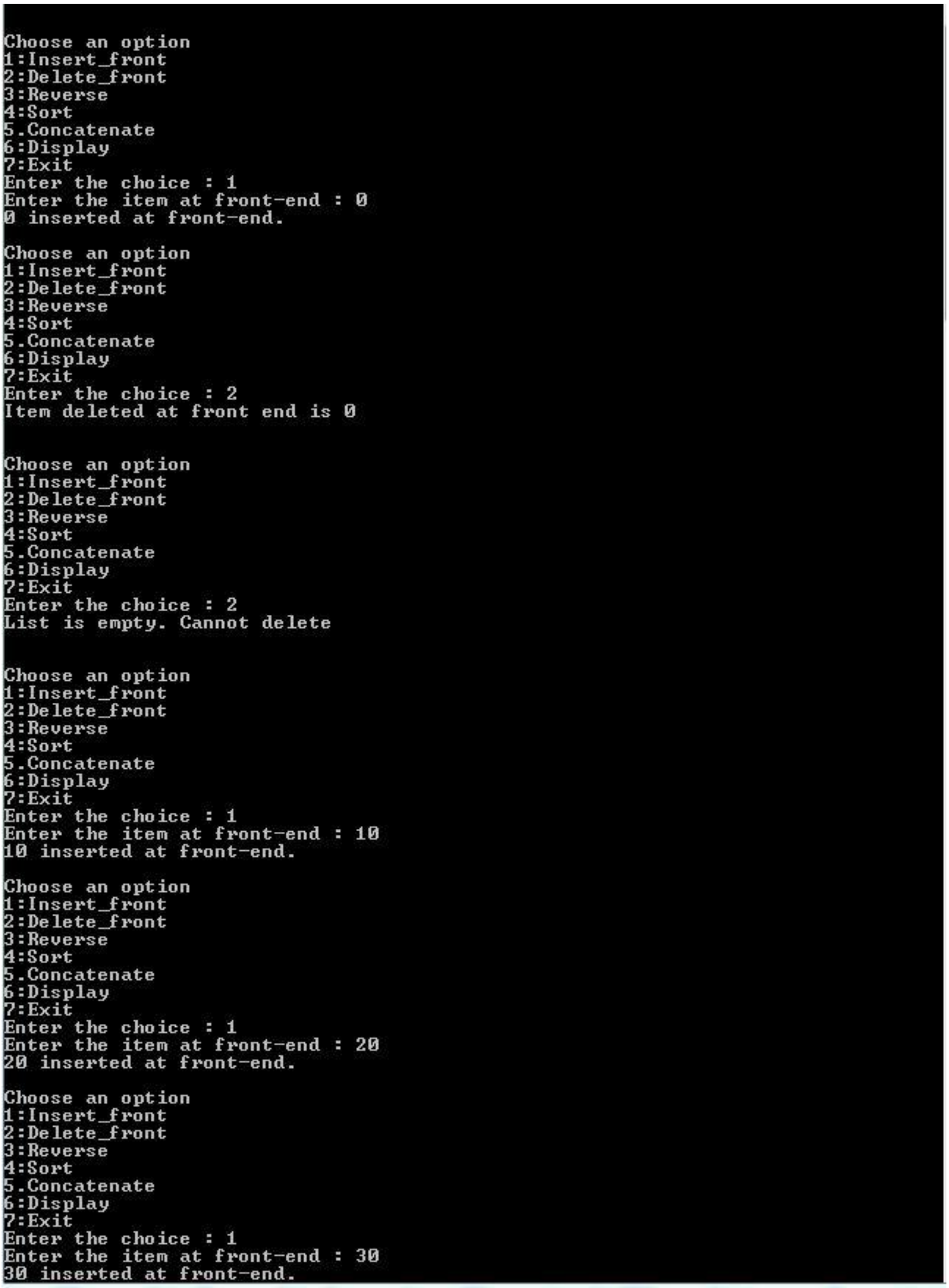
```
#include<stdio.h>
#include<conio.h>
#include<process.h>

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("\nMemory is full\n");
        exit(0);
    }
    return 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;
}

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 IF(NODE second,int item)
{
   NODE temp;
   temp=getnode();
   temp->info=item;
   temp->link=NULL;
   if(second==NULL)
      return temp;
   temp->link=second;
   second=temp;
   return second;
}

NODE IR(NODE second,int item)
{
   NODE temp,cur;
   temp=getnode();
   temp->info=item;
```

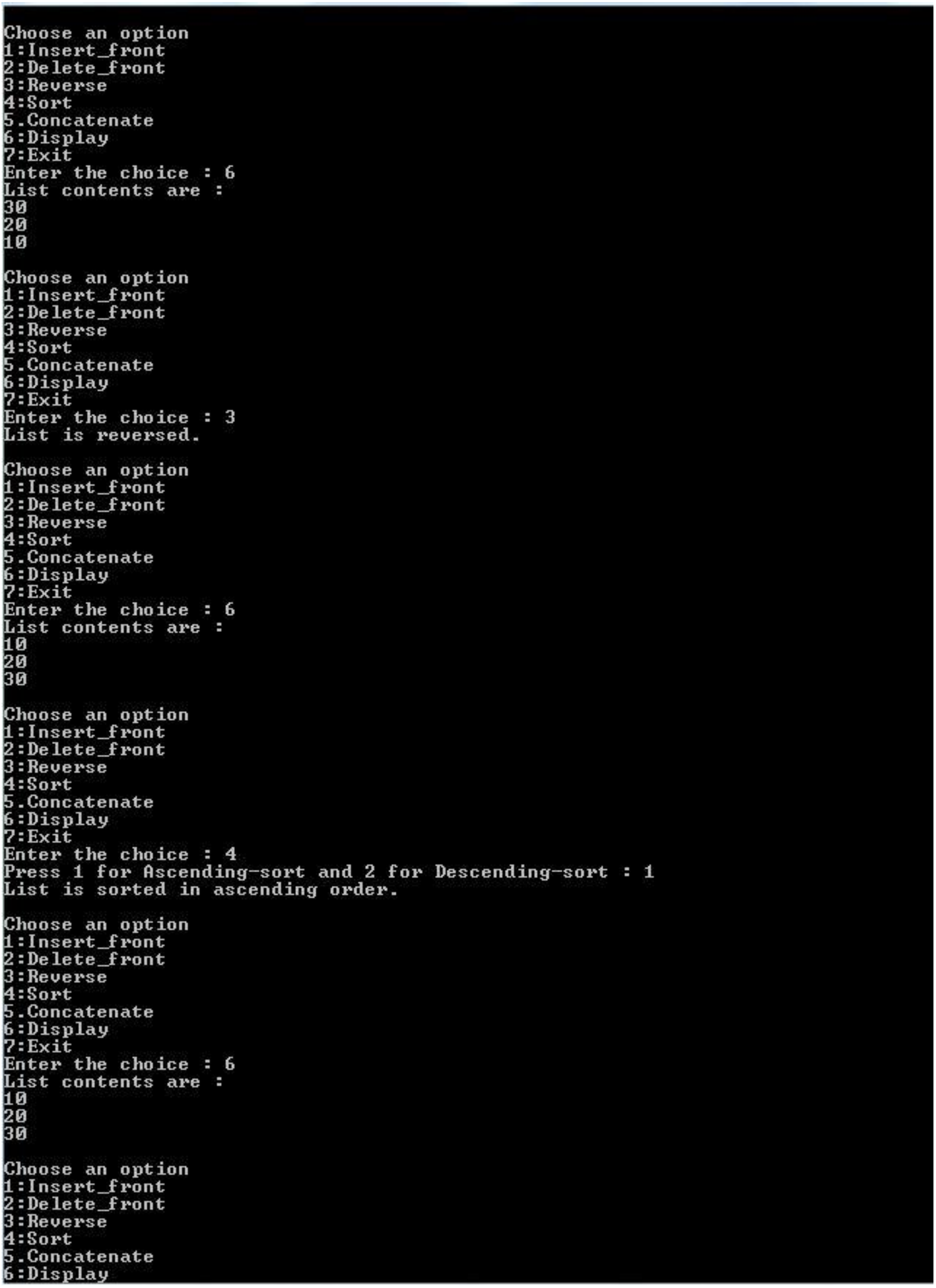
```
    temp->link=NULL;
    if(second==NULL)
        return temp;
    cur=second;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=temp;
    return second;
}

NODE reverse(NODE first)
{
    NODE cur,temp;
    cur=NULL;
    while(first!=NULL)
    {
        temp=first;
        first=first->link;
        temp->link=cur;
        cur=temp;
    }
    return cur;
}

NODE ascending(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
          int temp;
          if(first== NULL)
          {
             return 0;
          }
    else
    {
              while(prev!= NULL)
              {
                 cur = prev->link;
                 while(cur!= NULL)
                 {
               if(prev->info > cur->info)
               {
                     temp = prev->info;
                      prev->info = cur->info;
                      cur->info = temp;
                    }
                    cur = cur->link;
                 }
                 prev= prev->link;
             }
          }
          return first;
}

NODE descending(NODE first)
{
   NODE prev=first;
   NODE cur=NULL;
          int temp;
          if(first==NULL)
          {
             return 0;
          }
          else
          {
             while(prev!= NULL)
             {
                 cur = prev->link;
                 while(cur!= NULL)
                 {
               if(prev->info < cur->info)
               {
                  temp = prev->info;
                   prev->info = cur->info;
```

```
                    cur->info = temp;
                     }
                    cur = cur->link;
                  }
                  prev= prev->link;
              }
    }
    return first;
}

NODE concatenate(NODE first,NODE second)
{
    NODE cur;
    if(first==NULL)
        return second;
    if(second==NULL)
        return first;
    cur=first;
    while(cur->link!=NULL)
    {
        cur=cur->link;
    }
    cur->link=second;
    return first;
}

void display(NODE first)
{
    NODE temp;
    if(first==NULL)
    {
              printf("List is empty. Cannot display items.\n");
          return;
          }
          printf("List contents are : ");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("\n%d",temp->info);
    }
}

void main()
{
    int item,choice,pos,element,option,choice2,item1,num,i;
    NODE first=NULL;
    NODE second=NULL;
    for(;;)
    {
        printf("\n\nChoose an option");
        printf("\n1:Insert_front \n2:Delete_front \n3:Reverse \n4:Sort
\n5.Concatenate \n6:Display \n7:Exit\n");
        printf("Enter the choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the item at front-end : ");
                  scanf("%d",&item);
                  first=insert_front(first,item);
                  printf("%d inserted at front-end.",first->info);
                  break;
            case 2: first=delete_front(first);
                  break;
            case 3: first=reverse(first);
                   printf("List is reversed.");
                   break;
            case 4: printf("Press 1 for Ascending-sort and 2 for Descending-
sort : ");
                   scanf("%d",&option);
                   if(option==1)
                   {
                       first=ascending(first);
                       printf("List is sorted in ascending order.");
                   }
                   if(option==2)
```

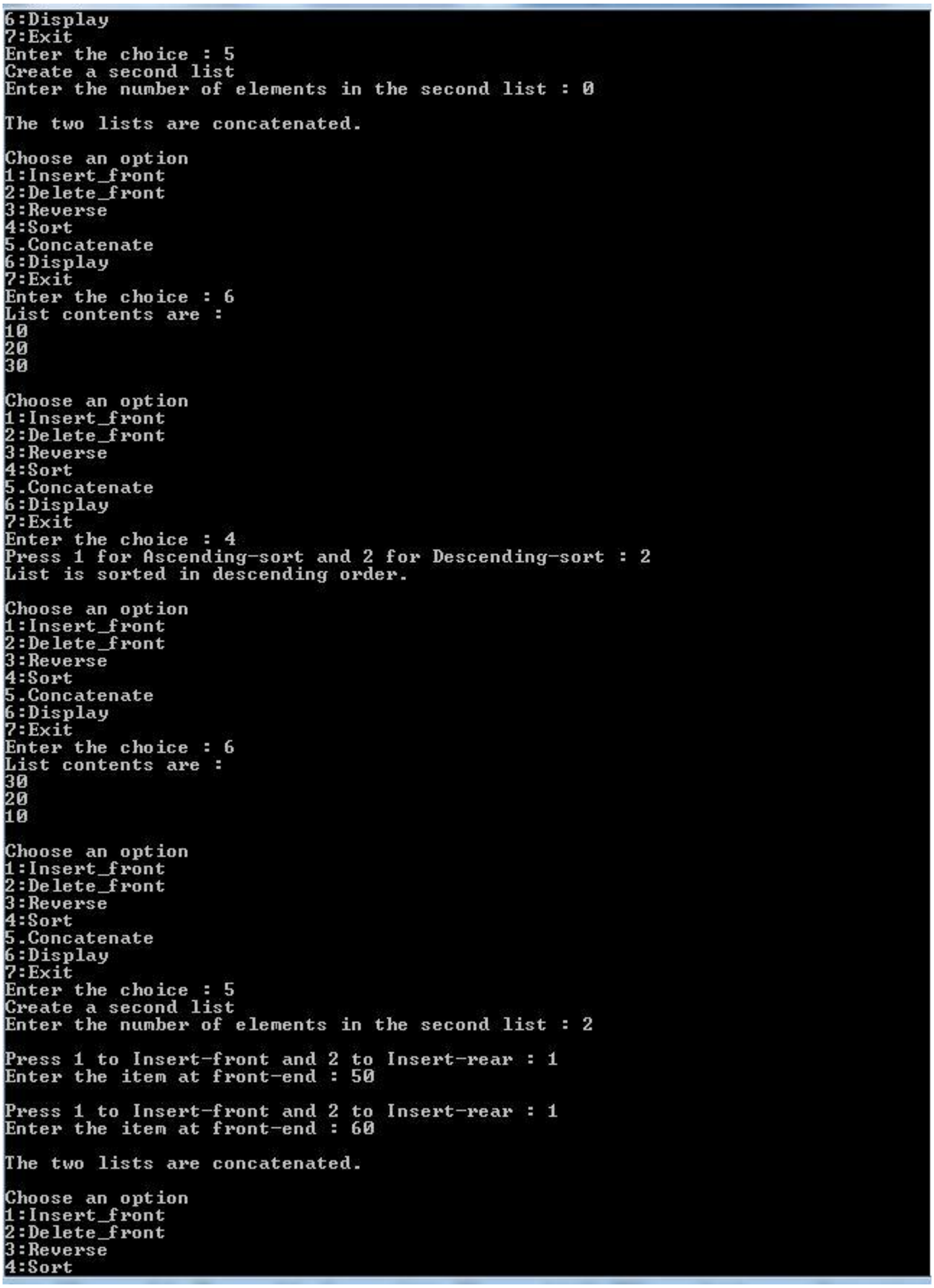
```
                    {
                       first=descending(first);
                       printf("List is sorted in descending order.");
                    }
                    break;
              case 5: printf("Create a second list\n");
                    printf("Enter the number of elements in the second list : ");
                    scanf("%d",&num);
                    for(i=1;i<=num;i++)
                    {
                       printf("\nPress 1 to Insert-front and 2 to Insert-rear : ");
                       scanf("%d",&choice2);
                       if(choice2==1)
                       {
                          printf("Enter the item at front-end : ");
                           scanf("%d",&item1);
                          second=IF(second,item1);
                       }
                       if(choice2==2)
                       {
                          printf("Enter the item at rear-end : ");
                           scanf("%d",&item1);
                          second=IR(second,item1);
                       }
                    }
                    first=concatenate(first,second);
                    printf("\nThe two lists are concatenated.");
                    break;
              case 6: display(first);
                   break;
              default:exit(0);
                   break;
           }
        }
     }
```



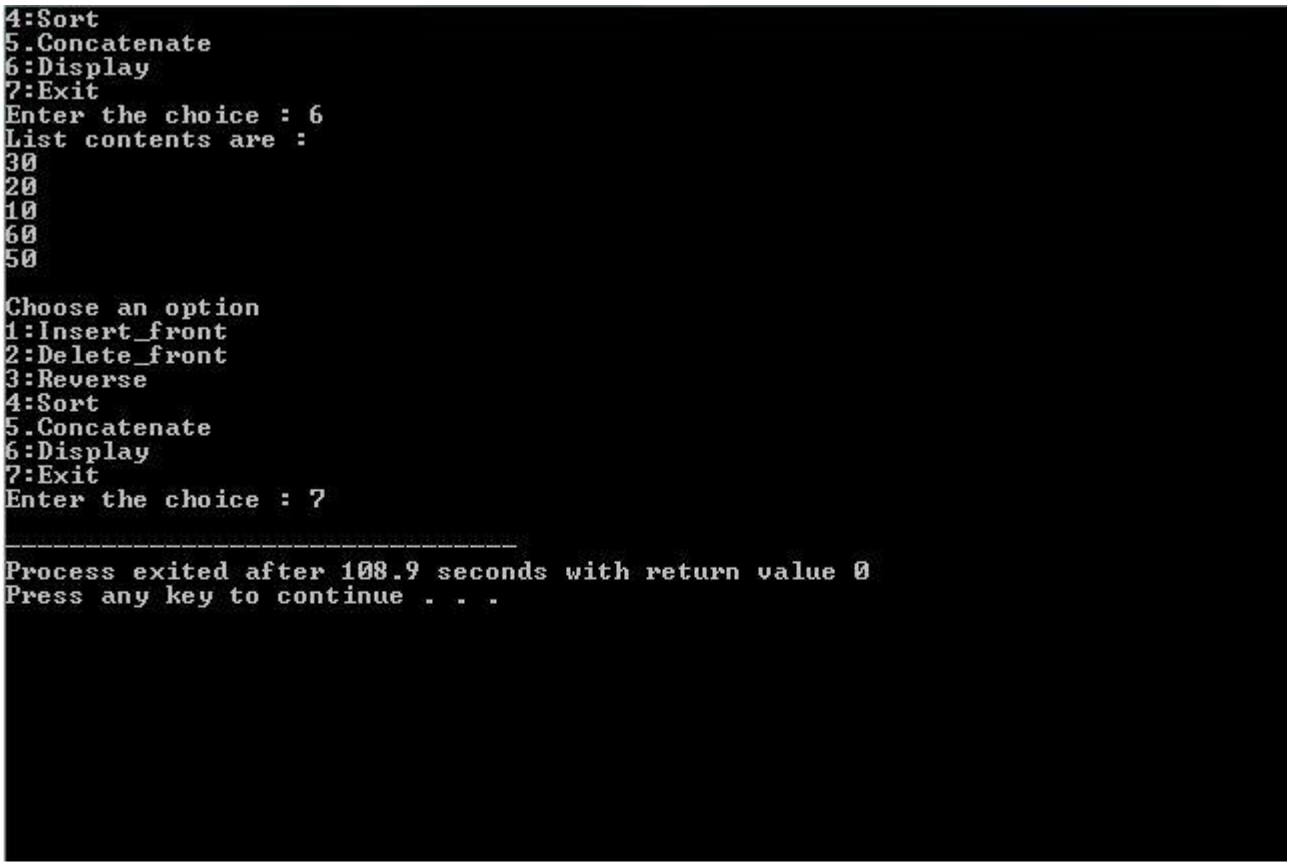
```
Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 0
0 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 2
Item deleted at front end is 0


Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 2
List is empty. Cannot delete


Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 10
10 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 20
20 inserted at front-end.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 1
Enter the item at front-end : 30
30 inserted at front-end.
```



```
Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
30
20
10

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 3
List is reversed.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
10
20
30

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 4
Press 1 for Ascending-sort and 2 for Descending-sort : 1
List is sorted in ascending order.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
10
20
30

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
```



```
6:Display
7:Exit
Enter the choice : 5
Create a second list
Enter the number of elements in the second list : 0

The two lists are concatenated.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
10
20
30

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 4
Press 1 for Ascending-sort and 2 for Descending-sort : 2
List is sorted in descending order.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
30
20
10

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 5
Create a second list
Enter the number of elements in the second list : 2

Press 1 to Insert-front and 2 to Insert-rear : 1
Enter the item at front-end : 50

Press 1 to Insert-front and 2 to Insert-rear : 1
Enter the item at front-end : 60

The two lists are concatenated.

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
```



```
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 6
List contents are :
30
20
10
60
50

Choose an option
1:Insert_front
2:Delete_front
3:Reverse
4:Sort
5.Concatenate
6:Display
7:Exit
Enter the choice : 7

--------------------------------
Process exited after 108.9 seconds with return value 0
Press any key to continue . . .
```