Lab-10 Singly Linked List

WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

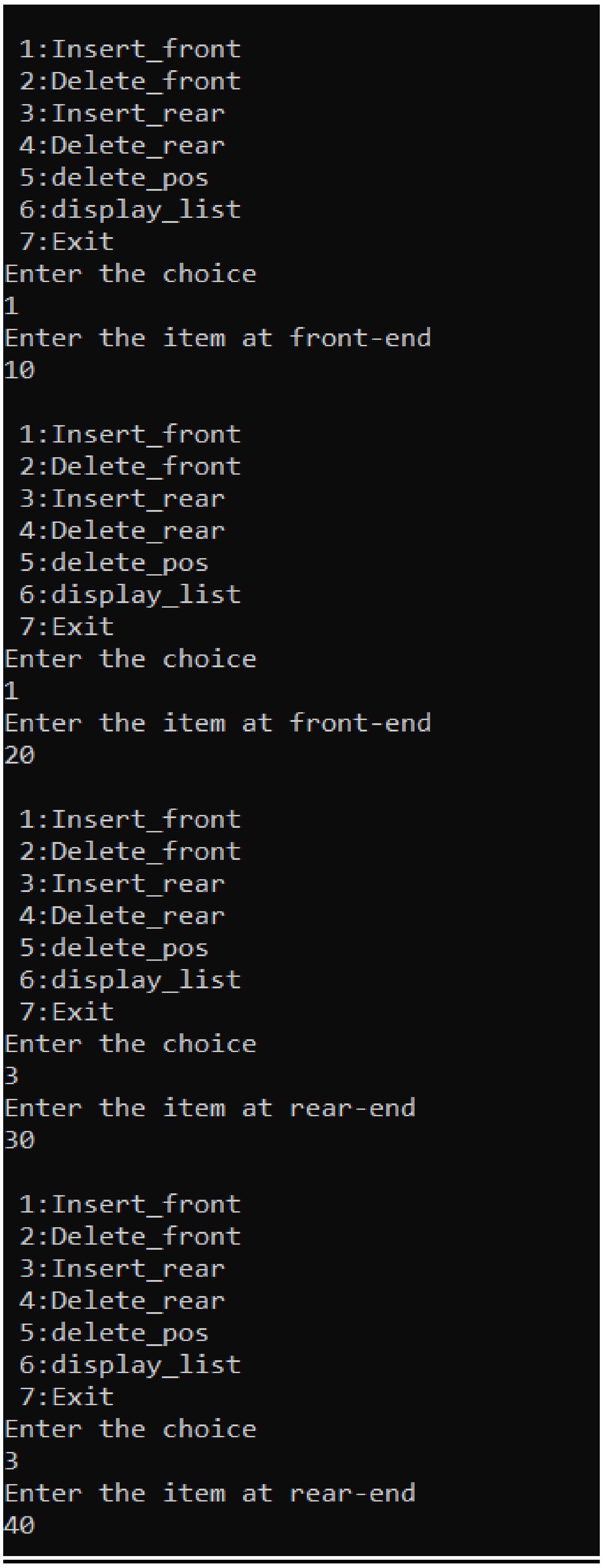
```
#include <stdio.h>
#include <conio.h>

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("mem 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;
}
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;
    }
    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("iten deleted at rear-end is %d", cur->info);
    free(cur);
    prev->link = NULL;
    return first;
}

NODE delete_pos(int pos, NODE first)
{
    NODE prev, cur;
    int count;
    if (first == NULL || pos <= 0)
    {
        printf("Invalid position\n");
        return NULL;
    }
    if (pos == 1)
    {
```

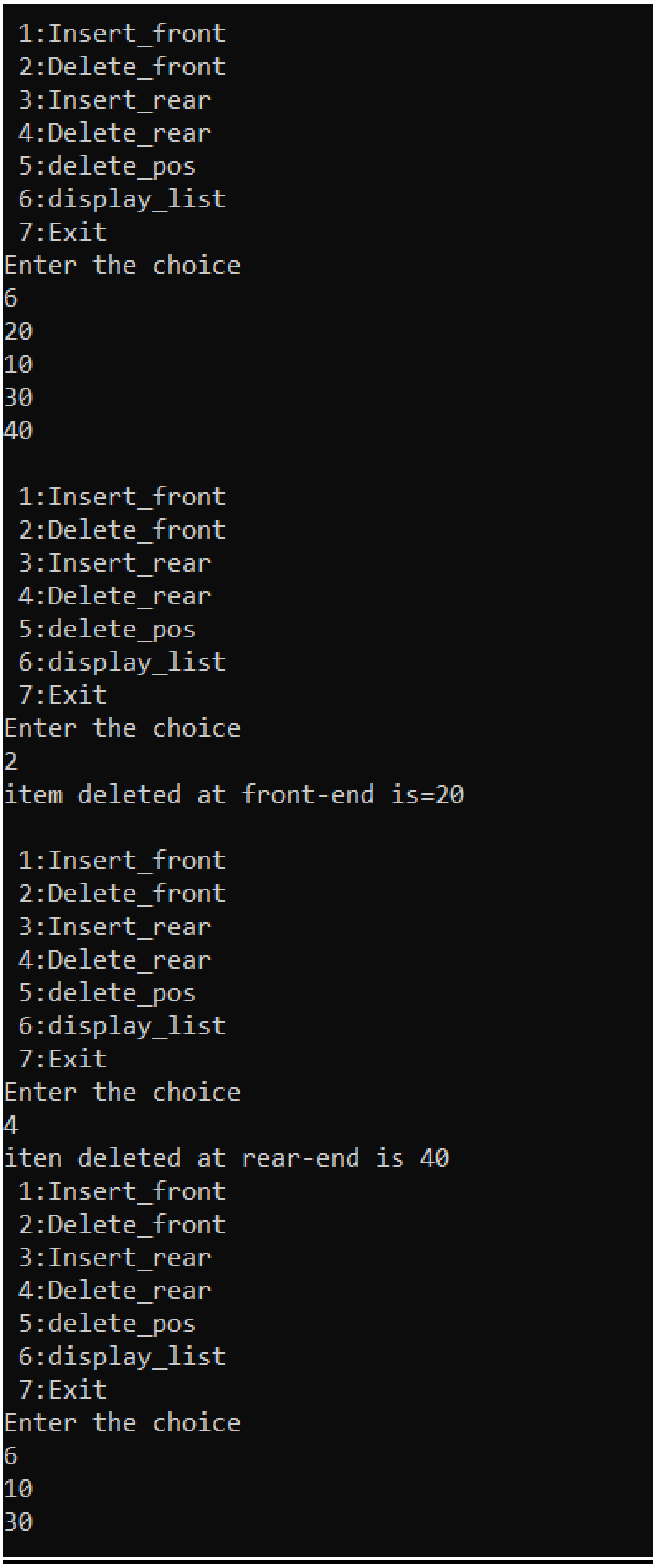
```
    cur = first;
    first = first->link;
    printf("iten deleted is %d", cur->info);
    freenode(cur);
    return first;
}
prev = NULL;
cur = first;
count = 1;
while (cur != NULL)
{
    if (count == pos)
    {
        break;
    }
    prev = cur;
    cur = cur->link;
    count++;
}
if (count != pos)
{
    printf("Invalid position\n");
    return first;
}
prev->link = cur->link;
printf("iten deleted is %d", cur->info);
freenode(cur);
```

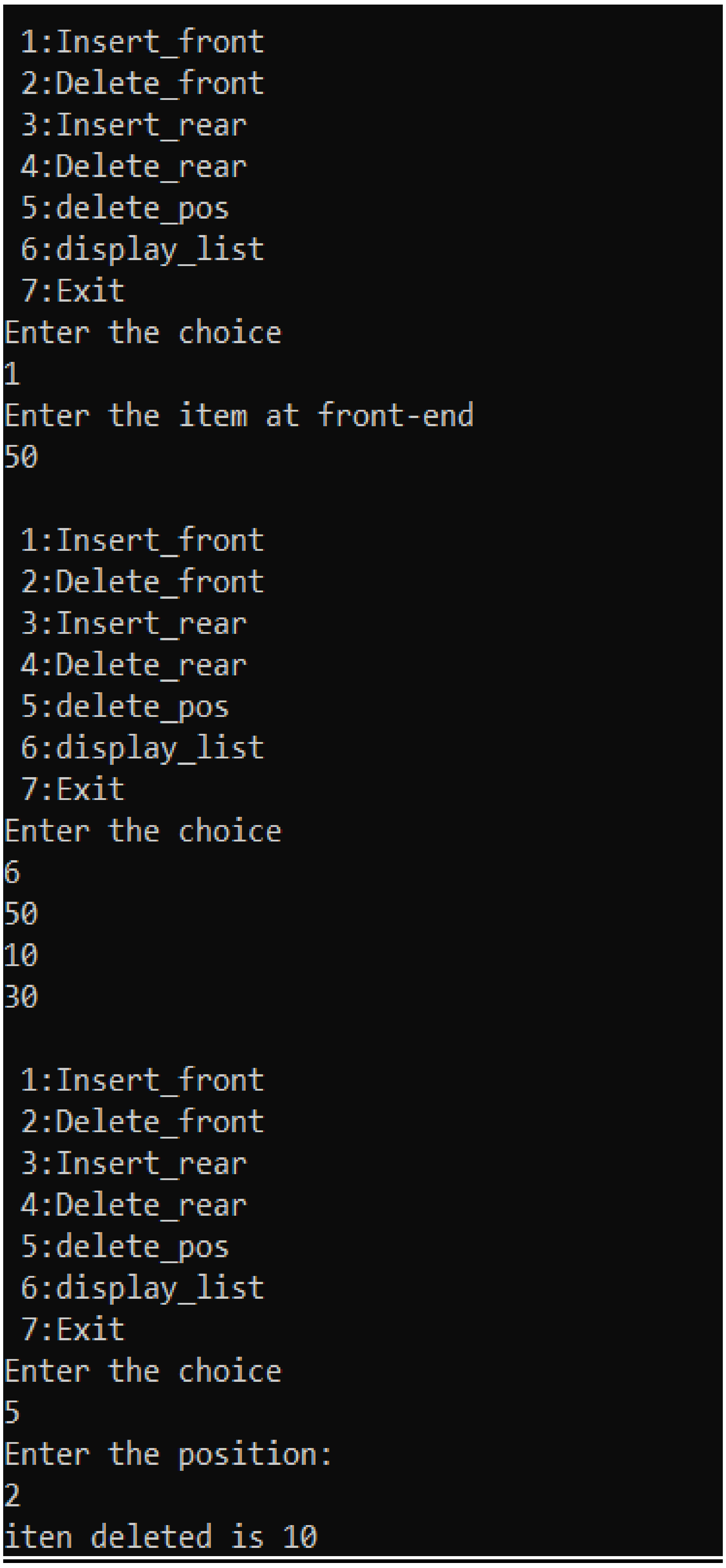
```
    return first;
}
void display(NODE first)
{
    NODE temp;
    if (first == NULL)
        printf("list empty cannot display items\n");
    for (temp = first; temp != NULL; temp = temp->link)
    {
        printf("%d\n", temp->info);
    }
}
void main()
{
    int item, choice, pos;
    NODE first = NULL;

    for (;;)
    {
        printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n
4:Delete_rear\n 5:delete_pos\n 6:display_list\n 7:Exit\n");
        printf("Enter the choice\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:
            printf("Enter the position:\n");
            scanf("%d", &pos);
            first = delete_pos(pos, first);
            break;
        case 6:
            display(first);
            break;
        default:
            exit(0);
            break;
        }
    }
}
```

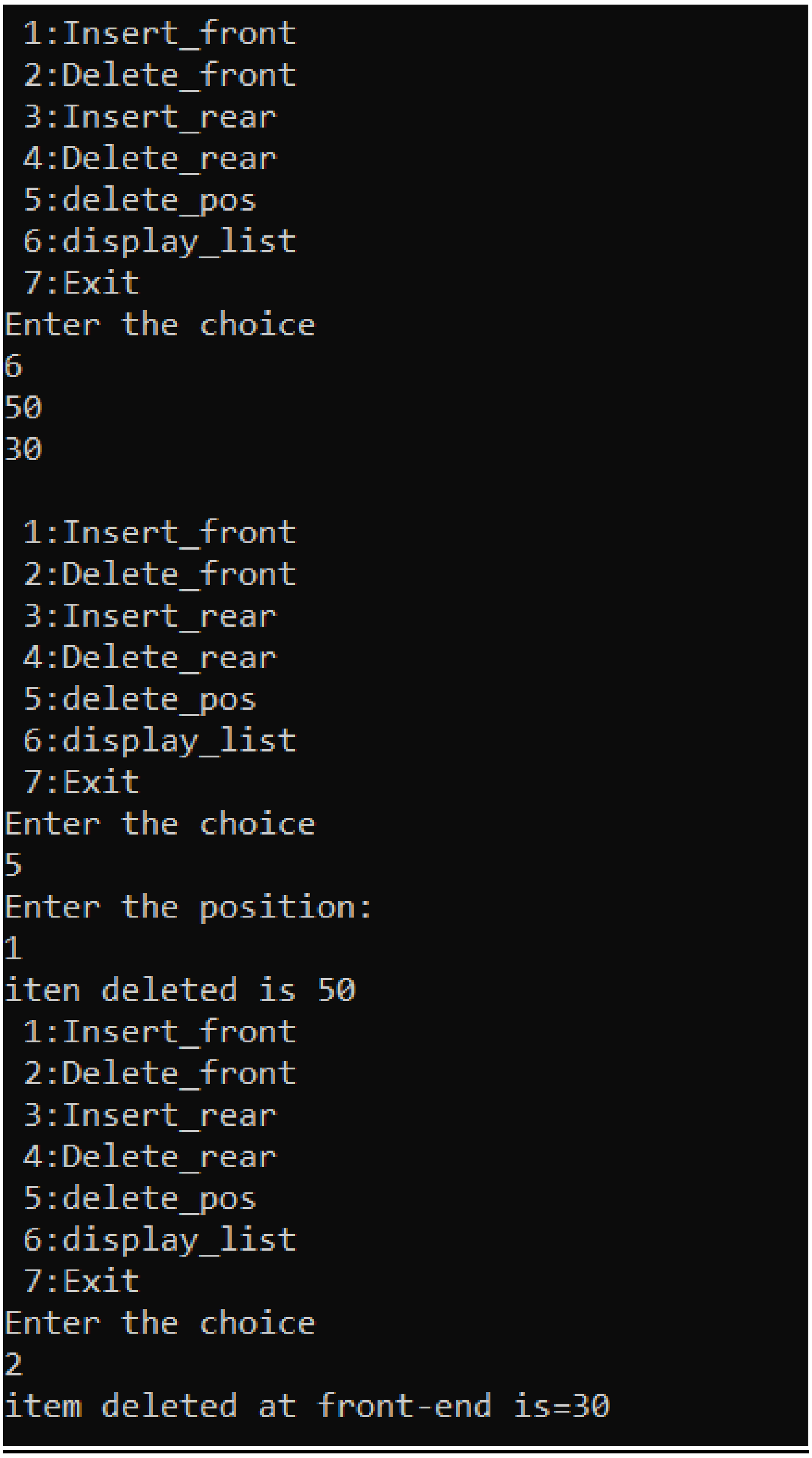
OUTPUT:



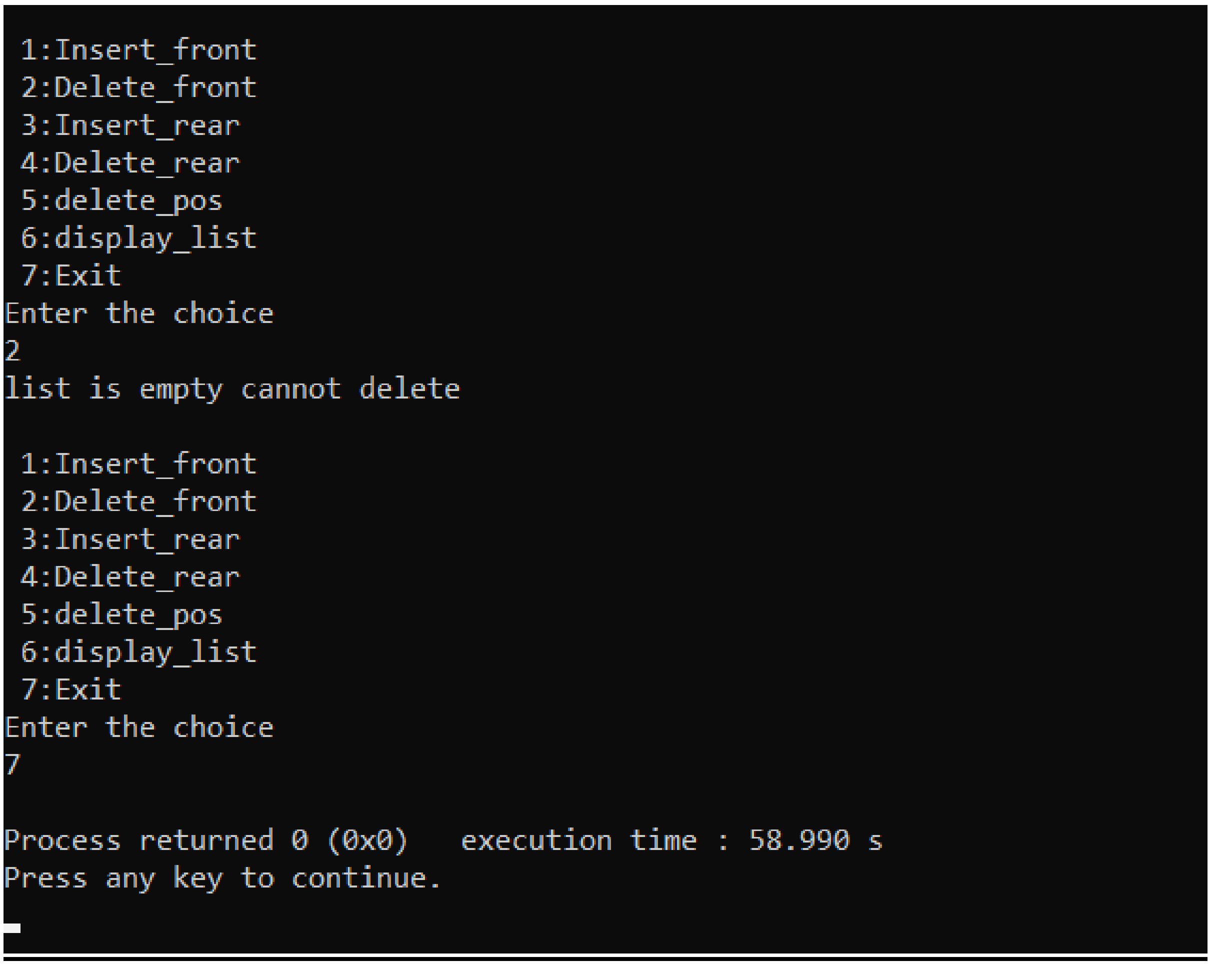
```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
1
Enter the item at front-end
10

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
1
Enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
3
Enter the item at rear-end
30

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
3
Enter the item at rear-end
40
```


```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
6
20
10
30
40

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
2
item deleted at front-end is=20

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
4
iten deleted at rear-end is 40
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
6
10
30
```


```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
1
Enter the item at front-end
50

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
6
50
10
30

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
5
Enter the position:
2
iten deleted is 10
```



```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
6
50
30

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
5
Enter the position:
1
iten deleted is 50
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
2
item deleted at front-end is=30
```



```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
2
list is empty cannot delete

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:delete_pos
 6:display_list
 7:Exit
Enter the choice
7

Process returned 0 (0x0)   execution time : 58.990 s
Press any key to continue.
```