WAP to Implement Singly Linked List with following operations a)Create a linked list. b)Insertion of a node at first position, at end of list. c)Display the contents of the linked list. d)Deletion of first element, last element in the list.

SOURCE CODE:

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

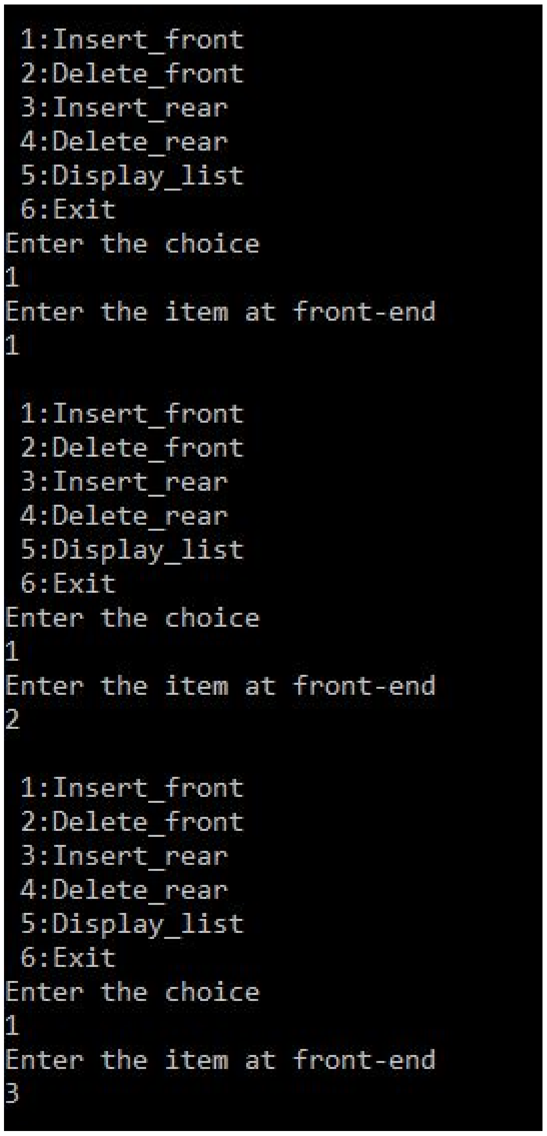
```
{
printf("List is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("Item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return 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("Item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List empty cannot display items\n");
 printf("Contents of the list:\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:Display_list\n
6: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:display(first);
            break;
            case 6:exit(0);
 default:printf("Invalid choice!\n");
            break;
 }
}

}
```

OUTPUT:

(Insertion at front-end)

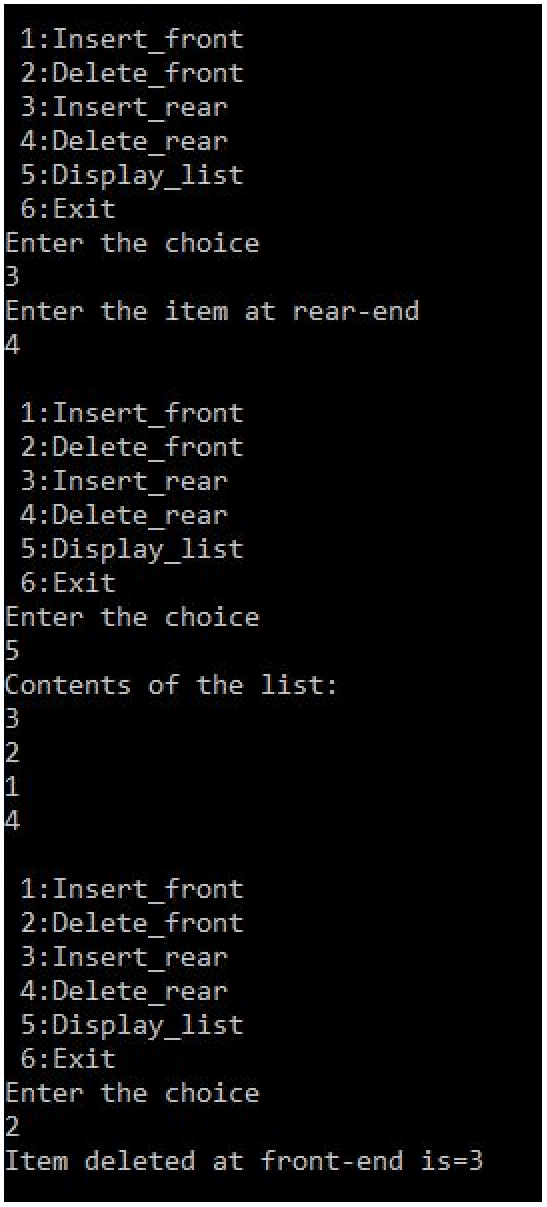


```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
1
Enter the item at front-end
1

 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
```

(Insertion at rear-end, display, deletion at front-end)

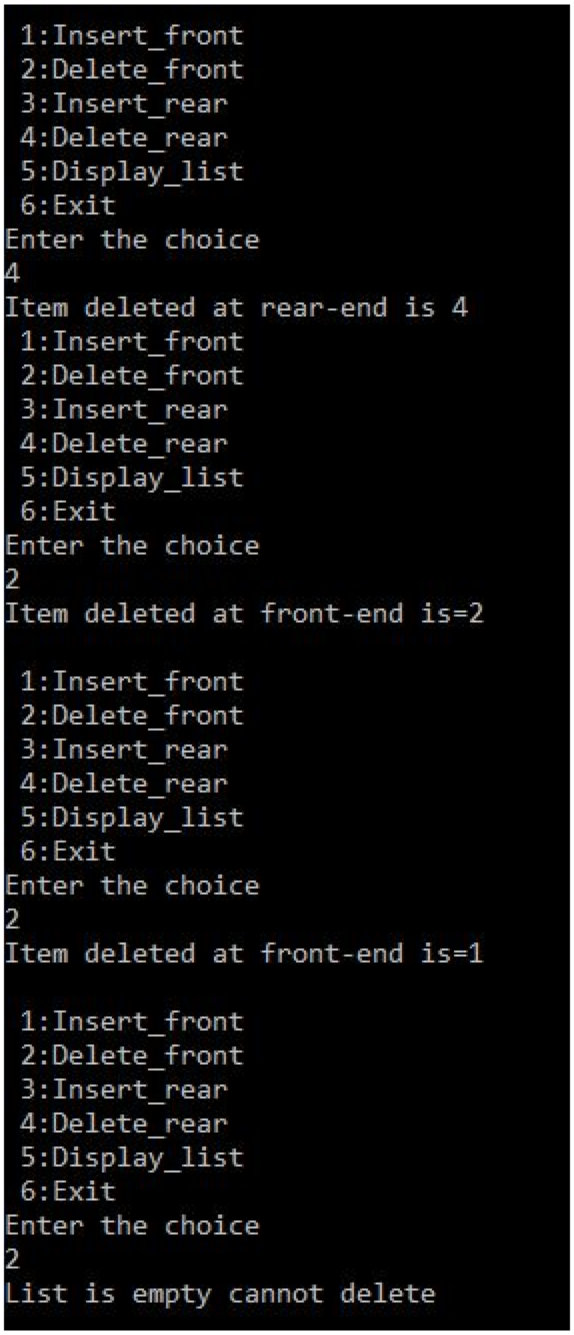


```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
3
Enter the item at rear-end
4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
5
Contents of the list:
3
2
1
4

 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
```

(deletion at rear-end, deletion at front-end, empty list condition)

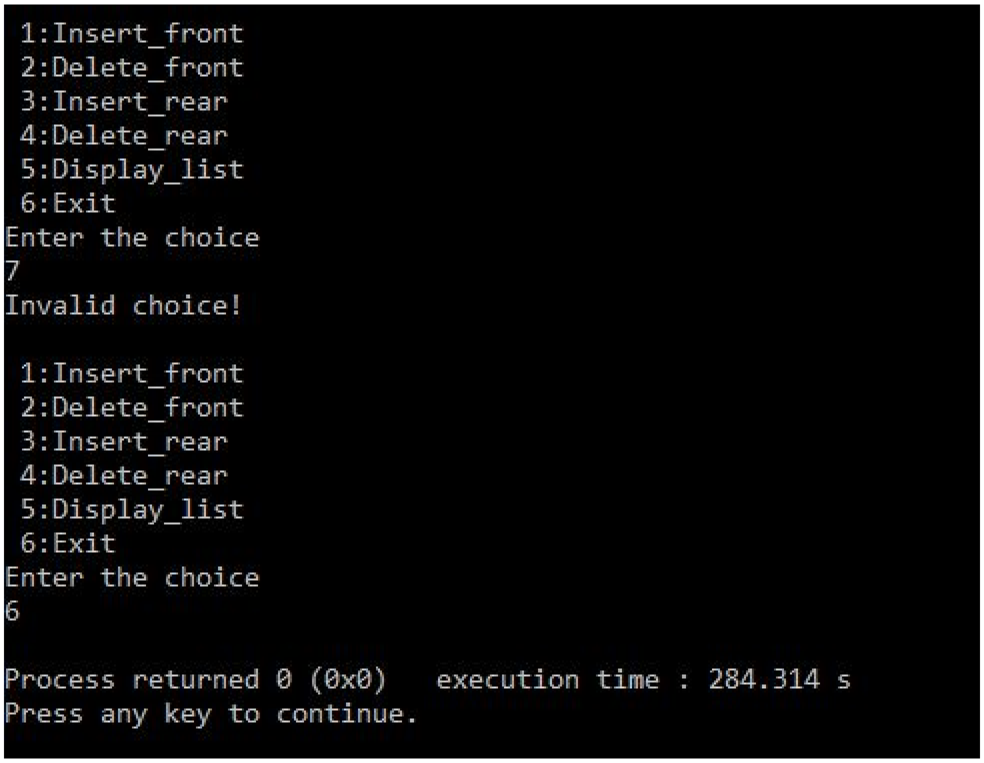


```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
4
Item deleted at rear-end is 4
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
2
Item deleted at front-end is=2

 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
2
List is empty cannot delete
```

(invalid case, exit case)



```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
7
Invalid choice!

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Display_list
 6:Exit
Enter the choice
6

Process returned 0 (0x0)   execution time : 284.314 s
Press any key to continue.
```