```
#include <stdio.h>
struct node {
  int info;
  struct node *link;
  };
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); }
NODE insert_front(NODE first, int item);
 {
   NODE temp = getnode();
   temp -> info = item;
   temp -> link = NULL;
   if (first == NULL)
     return temp;
     temp -> link = first;
```

```
return temp;
}
NODE delete_front(NODE first)
{
 NODE temp;
 if(first==NULL)
  {
   printf("List is empty\n");
   return first;
  }
   printf("Item deleted %d", (first->info));
   temp = first;
   temp = temp->link;
   free(first);
   return temp;
  }
NODE delete_rear(NODE first)
{
 NODE cur = first, prev = NULL;
 if(first == NULL)
 {
  printf("List empty\n");
  return NULL;
 }
 if(first->Link == NULL) {
  printf("Item deleted is %d\n", first->info);
   free(first)
   return NULL;
  }
  while(cur->link != NULL) {
   prev = cur;
   cur = cur->link; }
```

```
printf("Item deleted is %d \n", cur->info);
free(cur);
prev->link = NULL;
return first;
}
NODE del_pos(NODE first, int pos)
{
NODE cur, prev;
int c=1;
if(first == NULL || pos<0)
{
printf("Invalid position \n");
return NULL;
}
if(pos == 1)
{
free(first);
return NULL;
}
cur = first;
prev = NULL;
while(cur != NULL)
{
if(c == pos)
{
printf("Element deleted is %d", cur->info);
prev->link = cur->link;
free(cur);
return first;
}
prev = cur;
cur = cur->link;
```

```
        ptr = ptr->link;
        pos++;
    }
    printf("Element not found\n");
    return first;
}
void display(NODE first)
{
    if(first == NULL)
    {
        printf("List is empty\n");
        return;
    }
    printf("Elements of the list are:\n");
    for(NODE ptr=first; ptr!=NULL; ptr=ptr->link)
        printf("%d\n",ptr->info);
}
int main()
{
    int item, ch, pos;
    NODE first = NULL;
    for(;;)
    {
        printf("\n1.Insert\n2.delete front\n3.delete rear\n
            4.delete pos\n5.Display\n6.Exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("Enter element to be inserted\n");
                    scanf("%d",&item);
                    first = insert_front(first,item);
                    break;
            case 2: first = delete_front(first);
```

```
        break;
case 3: first = delete_rear(first);
        break;
case 4: printf("Enter position \n");
        scanf("%d", &pos);
        first = delete_pos(first, pos);
        break;
case 5: display(first);
        break;
default: return 0;
    }
  }
}
```

```c
#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(NODE));//
 if(x==NULL)
 {
  printf("memory full \n");
  exit(0);
 }
 return x;
}

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

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

NODE delete_front(NODE first)
{
 NODE temp;
 if(first == NULL)
 {
  printf("List is empty\n");
  return first;
 }
 printf("Item deleted %d",(first->info));
 temp = first;
 temp=temp->link;
 free(first);
 return temp;
}

NODE delete_rear(NODE first)
{
 NODE cur=first,prev=NULL;
```

```
 54   if(first==NULL)
 55   {
 56     printf("List empty\n");
 57     return NULL;
 58   }
 59   if(first->link==NULL)
 60   {
 61     printf("item deleted is
%d\n",first->info);
 62     free(first);
 63     return NULL;
 64   }
 65   while(cur->link!=NULL)
 66   {
 67     prev=cur;
 68     cur=cur->link;
 69   }
 70   printf("Item deleted is %d\n",(cur->info));
 71   free(cur);
 72   prev->link=NULL;
 73   return first;
 74 }
 75
 76 void display(NODE first)
 77 {
 78   if(first==NULL)
 79   {
 80     printf("List is empty\n");
 81     return;
 82   }
 83   printf("Elements of the list are : \n");
 84   for(NODE i=first;i!=NULL;i=i->link)
 85   printf("%d\n",i->info);
 86 }
 87
 88 NODE delete_pos(NODE first,int pos)
 89 {
 90   NODE cur,prev;
 91   int c=1;
 92   if(first==NULL||pos<0)
 93   {
 94     printf("Invalid position\n");
 95     return NULL;
 96   }
 97   if(pos==1)
 98   {
 99     free(first);
100     return NULL;
101   }
102   cur=first;
103   prev=NULL;
104   while(cur!=NULL)
105   {
```

```
105  {
106    if(c==pos)
107    {
108      printf("Element deleted is
%d",cur->info);
109      prev->link=cur->link;
110      free(cur);
111      return first;
112    }
113    prev=cur;
114    cur=cur->link;
115    c++;
116  }
117  printf("Element not found\n");
118  return first;
119 }
120
121
122
123
124 int main()
125 {
126   int item,ch,pos;
127   NODE first=NULL;
128   for(;;)
129   {
130     printf("\n1.Insert front\n2.Delete
front\n3.Delete
rear\n4.Delete_pos\n5.Display\n");
131     scanf("%d",&ch);
132     switch(ch)
133     {
134       case 1:
135         printf("Enter element to be
inserted\n");
136         scanf("%d",&item);
137         first = insert_front(first,item);
138         break;
139       case 2:
140         first = delete_front(first);
141         break;
142       case 3:
143         first = delete_rear(first);
144         break;
145       case 4:
```

```
146         printf("Enter position\n");
147         scanf("%d",&pos);
148         first = delete_pos(first,pos);
149         break;
150       case 5:
151         display(first);
152         break;
153       default:return 0;
154     }
155   }
156 }
```

```
1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
1
Enter element to be inserted
10

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
1
Enter element to be inserted
20

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
1
Enter element to be inserted
30

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
1
Elements of the list are :
40
30
20
10

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
2
Item deleted 40
1.Insert front
2.Delete front
```

```
2
Item deleted 40
1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
5
Elements of the list are :
30
20
10

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
3
Item deleted is 10

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
5
Elements of the list are :
30
20

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
4
Enter position
2
Element deleted is 20
1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
5
Elements of the list are :
30
```