```c
#include<stdio.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)
```

```
31  return temp;
32  temp->link=first;
33  first=temp;
34  return first;
35  }
36  NODE insert_rear(NODE first,int item)
37  {
38  NODE temp,cur;
39  temp=getnode();
40  temp->info=item;
41  temp->link=NULL;
42  if(first==NULL)
43  return temp;
44  cur=first;
45  while(cur->link!=NULL)
46  cur=cur->link;
47  cur->link=temp;
48  return first;
49  }
50  NODE insert_pos(int item,int pos,NODE first)
51  {
52  NODE temp,cur,prev;
53  int count;
54  temp=getnode();
55  temp->info=item;
56  temp->link=NULL;
57  if(first==NULL&&pos==1)
58  {
59  return temp;
60  }
```

```
61  if(first==NULL)
62  {
63  printf("invalid position\n");
64  return first;
65  }
66  if(pos==1)
67  {
68  temp->link=first;
69  first=temp;
70  return temp;
71  }
72  count=1;
73  prev=NULL;
74  cur=first;
75  while(cur!=NULL&&count!=pos)
76  {
77  prev=cur;
78  cur=cur->link;
79  count++;
80  }
81  if(count==pos)
82  {
83
84  prev->link=temp;
85  temp->link=cur;
86  return first;
87  }
88  printf("invalid position\n");
89  return first;
90  }
```

```
91  NODE delete_front(NODE first)
92  {
93  NODE temp;
94  if(first==NULL)
95  {
96  printf("CANNOT DELETE AS LIST IS EMPTY\n");
97  return first;
98  }
99  temp=first;
100 temp=temp->link;
101 printf("ITEM DELETED AT FRONT END=%d\n",first->info);
102 free(first);
103 return temp;
104 }
105 NODE delete_rear(NODE first)
106 {
107 NODE cur,prev;
108 if(first==NULL)
109 {
110 printf("CANNOT DELETE AS LIST IS EMPTY\n");
111 return first;
112 }
113 if(first->link==NULL)
114 {
115 printf("ITEM DELETED=%d\n",first->info);
116 free(first);
117 return NULL;
118 }
119 prev=NULL;
120 cur=first;
```

```
121  while(cur->link!=NULL)
122  {
123  prev=cur;
124  cur=cur->link;
125  }
126  printf("ITEM DELETED AT REAR END=%d\n",cur->info);
127  free(cur);
128  prev->link=NULL;
129  return first;
130  }
131  NODE delete_pos(int pos,NODE first)
132  {
133  NODE cur;
134  NODE prev;
135  int count,flag=0;
136  if(first==NULL || pos<0)
137  {
138  printf("invalid position\n");
139  return NULL;
140  }
141  if(pos==1)
142  {
143  cur=first;
144  first=first->link;
145  freenode(cur);
146  return first;
147  }
148  prev=NULL;
149  cur=first;
150  count=1;
```

```
151  while(cur!=NULL)
152  {
153  if(count==pos)
154  {
155  flag=1;
156  break;
157  }
158  count++;
159  prev=cur;
160  cur=cur->link;
161  }
162  if(flag==0)
163  {
164  printf("invalid position\n");
165  return first;
166  }
167  printf("ITEM DELETED AT POSITION %d is %d\n",pos,cur->info);
168  prev->link=cur->link;
169  freenode(cur);
170  return first;
171  }
172  void display(NODE first)
173  {
174  NODE temp;
175  if(first==NULL)
176  printf("list empty cannot display items\n");
177  for(temp=first;temp!=NULL;temp=temp->link)
178  {
179  printf("%d\n",temp->info);
180  }
```

```
181 }
182 void main()
183 {
184 int item,choice,pos;
185 NODE first=NULL;
186 for(;;)
187 {
188 printf("1.Insert_front\n2.Insert_rear\n3.Insert at given Position\n4.Delete Front\n5.Delete Rear\n6.Delete at a given position\n7.Display the list\n8.Exit\n"
189 printf("enter the choice\n");
190 scanf("%d",&choice);
191 switch(choice)
192 {
193 case 1:printf("enter the item at front-end\n");
194        scanf("%d",&item);
195        first=insert_front(first,item);
196        break;
197 case 2:printf("enter the item at rear-end\n");
198        scanf("%d",&item);
199        first=insert_rear(first,item);
200        break;
201 case 3:printf("enter the item to be inserted at given position\n");
202        scanf("%d",&item);
203        printf("enter the position\n");
204        scanf("%d",&pos);
205        first=insert_pos(item,pos,first);
206        break;
207 case 4:first=delete_front(first);
208        break;
209 case 5:first=delete_rear(first);
210        break;
```

```
210             break;
211     case 6:printf("Enter the position\n");
212             scanf("%d",&pos);
213             first=delete_pos(pos,first);
214             break;
215     case 7:display(first);
216             break;
217     default:exit(0);
218              break;
219     }
220     }
221     }
```

Select "C:\Users\Neha Chadaga\Desktop\linkedlist2Lab.exe"

```
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
1
enter the item at front-end
12
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
2
enter the item at rear-end
67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
3
enter the item to be inserted at given position
78
enter the position
2
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
7
12
78
```

```
12
78
67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
1
enter the item at front-end
22
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
1
enter the item at front-end
66
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
7
66
22
12
78
67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
```

```
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
4
ITEM DELETED AT FRONT END=66
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
5
ITEM DELETED AT REAR END=67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
6
Enter the position
2
ITEM DELETED AT POSITION 2 is 12
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
8

Process returned 0 (0x0)   execution time : 39.875 s
Press any key to continue.
```