

→ DS Lab Program 6

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

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

```
struct Node {
```

```
    int info;
```

```
    struct Node *link;
```

```
};
```

```
typedef struct Node node;
```

```
node getnode () {
```

```
    node n;
```

```
    n = (node)malloc (sizeof (node));
```

```
    if (n == NULL) {
```

```
        printf ("Memory full \n");
```

```
        exit (0);
```

```
}
```

```
return n;
```

```
}
```

```
void freeNode (node n) {
```

```
    free (n);
```

```
}
```

```
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");
```

returns first;

y

printf ("Item deleted: %d\n", first->info);

temp = first;

temp = temp -> link;

free (first);

returns temp;

y

node delete - never (node first) {

node cur = first, prev = NULL;

if (first == NULL) {

printf ("List empty\n");

returns NULL;

y

if (first -> link == NULL) {

printf ("Item deleted: %d\n", first->info);

free (first);

returns NULL;

y

while (cur -> link != NULL) {

prev = cur;

cur = cur -> link;

y

printf ("Item deleted: %d\n", cur->info);

free (cur);

prev -> link = NULL;

returns first;

y

node delete_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 : %d\n", cur->info);

prev->link = cur->link;

free (cur);

return first;

}

prev = cur;

cur = cur->link;

c++;

}

printf ("Element not found (%d);\n", c);

return first;

}

int main () {

int item, ch, pos;

node first = NULL;

for (;;) {

printf ("1. Insert front in 2. Delete front
 in 3. Delete node in 4. Delete pos (ins.display(v));

scanf ("%d", &ch);

switch (ch) {

case 1: printf ("Enter element to insert (%d);\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: exit (0);

y

y

y

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 struct node{
4     int info;
5     struct node *link;
6 };
7 typedef struct node *NODE;
8 NODE getnode()
9 {
10     NODE x;
11     x=(NODE)malloc(sizeof(NODE));//
12     if(x==NULL)
13     {
14         printf("memory full \n");
15         exit(0);
16     }
17     return x;
18 }
19
20 void freenode(NODE x)
21 {
22     free(x);
23 }
24
25 NODE insert_front(NODE first,int item)
26 {
27     NODE temp = getnode();
28     temp->info = item;
29     temp->link = NULL;
30     if(first == NULL)
31     {
32         return temp;
33     }
34     temp->link=first;
35     return temp;
36 }
37
38 NODE delete_front(NODE first)
39 {
40     NODE temp;
41     if(first == NULL)
42     {
43         printf("List is empty\n");
44         return first;
45     }
46     printf("Item deleted %d", (first->info));
47     temp = first;
48 }
```

```

43     }
44     printf("Item deleted %d", (first->info));
45     temp = first;
46     temp=temp->link;
47     free(first);
48     return temp;
49 }
50
51 NODE delete_rear(NODE first)
52 {
53     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 }

```

```
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    {
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 int main()
121 {
122     int item,ch,pos;
123     NODE first=NULL;
124     for(;;)
125     {
126         printf("\n1.Insert front\n2.Delete front\n3.Delete rear\n4.Delete_pos\n5.Display\n");
127         scanf("%d",&ch);
```

```
1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
1
Enter element to be inserted
50

1.Insert front
2.Delete front
3.Delete rear
4.Delete_pos
5.Display
1
Enter element to be inserted
40

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
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
10

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

```
3.Delete rear  
4.Delete_pos  
5.Display  
5  
Elements of the list are :  
10  
20  
30  
40  
50
```

```
1.Insert front  
2.Delete front  
3.Delete rear  
4.Delete_pos  
5.Display  
2  
Item deleted 10  
1.Insert front  
2.Delete front  
3.Delete rear  
4.Delete_pos  
5.Display  
3  
Item deleted is 50
```

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

```
1.Insert front  
2.Delete front  
3.Delete rear  
4.Delete_pos  
5.Display  
4  
Enter position  
2  
Element deleted is 30  
1.Insert front  
2.Delete front  
3.Delete rear  
4.Delete_pos  
5.Display
```