→ DS Lab Program 6

```
#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\n", 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: %d\n", first->info);
    free(first);
    return NULL;
}
while (cur -> link != NULL) {
    prev = cur;
    cur = cur -> link;
}
printf("Item deleted: %d\n", cur->info);
free(cur);
prev -> link = NULL;
return first;
}
node delete_pos(node first, int pos) {
node cur, prev;
int i = 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 \n");
return first;
}
int main () {
int item, ch, pos;
node first = NULL;
for (;;) {
printf ("\n1 Insert front \n2. Delete front
    \n3. Delete rear \n4. Delete pos \n5. display\n");
scanf ("%d", &ch);
switch (ch) {
case 1: printf ("Enter element to insert \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);
    }
}
}
```

```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;
```

```
    }
    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;
}

void display(NODE first)
{
    if(first==NULL)
    {
        printf("List is empty\n");
        return;
    }
    printf("Elements of the list are : \n");
    for(NODE i=first;i!=NULL;i=i->link)
    printf("%d\n",i->info);
}
```

```
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
```