

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

```
#include <conio.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 insert-rear (NODE first, int item)
{

```

```

    NODE temp = getnode();

```

```

    cur;

```

```

    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 insert-pos (NODE first, int item, int pos)
{

```

```

    int c = 1;

```

```

    NODE temp = getnode(); cur, prev;

```

```

    temp->info = item;

```

```

    if (pos == 1)

```

```

    {

```

```

        temp->link = first;

```

```

        return temp;

```

```

    }

```

```

    cur = first;

```

```

    prev = NULL;

```

```

    while (cur != NULL) {

```

```
if (pos == c)
```

```
{
```

```
prev->link = temp;
```

```
temp->link = cur;
```

```
return first;
```

```
}
```

```
c++;
```

```
prev = cur;
```

```
cur = cur->link;
```

```
}
```

```
printf("Invalid position \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 i = first; i != NULL; i = i->link)
```

```
printf("%d \n", i->info);
```

```
}
```

```
int main ()
```

```
{
```

```
int item, ch, pos;
```

```
NODE first = NULL;
```

```
for (;;) 
```

```
{
```

```
printf("\n 1. Insert front \n 2. Insert rear \n 3.
```

```
Insert position \n 4. display \n 5. 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: printf("Enter element to be inserted \n");
```

```
scanf("%d", &item);
```

```
first = insert-rear(first, item);
```

```
break;
```

```
case 3: printf("Enter element to be inserted \n");
```

```
scanf("%d", &item);
```

```
printf("Enter position \n");
```

```
scanf("%d", &pos);
```

```
first = insert-pos(first, item, pos);
```

```
break;
```

```
case 4: display(first);
```

```
break;
```

```
default: return 0;
```

```
}
```

```
}
```

```
}
```



```

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         return temp;
32     temp->link=first;
33     return temp;
34 }
35
36 NODE insert_rear(NODE first,int item)
37 {
38     NODE temp = getnode(),cur;
39     temp->info=item;
40     temp->link = NULL;
41     if(first==NULL)
42         return temp;
43     cur=first;
44     while(cur->link!=NULL)
45         cur=cur->link;
46     cur->link=temp;
47     return first;
48 }
49 void insert_at(NODE first,int item) {
50     printf("Enter position after which to
enter\n");
51     int pos;
52     scanf("%d",&pos);

```

```

53  NODE bef=getnode();
54  bef=first;
55  NODE nxt=getnode();
56  NODE temp=getnode();
57  temp->info=item;
58  temp->link=NULL;
59  for(int i=1;i<pos;i++)
60      bef=bef->link;
61  nxt=bef->link;
62  bef->link=temp;
63  temp->link=nxt;
64  printf("%d",temp->link->info);
65  }
66
67  void display(NODE first)
68  {
69      if(first==NULL)
70      {
71          printf("List is empty\n");
72          return;
73      }
74      printf("Elements of the list are : \n");
75      for(NODE i=first;i!=NULL;i=i->link)
76          printf("%d\n",i->info);
77  }
78
79  int main()
80  {
81      int item,ch;
82      NODE first=NULL;
83      for(;;)
84      {
85          printf("\n1.Insert front\n2.Insert At:
86          \n3.Insert rear\n4.Display\n");
87          scanf("%d",&ch);
88          switch(ch)
89          {
90              case 1:
91                  printf("Enter element to be
92                  inserted\n");
93                  scanf("%d",&item);
94                  first = insert_front(first,item);
95                  break;
96              case 2:
97                  printf("Enter element to be
98                  inserted\n");
99                  scanf("%d",&item);
100                 insert_at(first,item);
101                 break;
102              case 3:
103                 printf("Enter element to be
104                 inserted\n");
105                 scanf("%d",&item);

```

```
102     first = insert_rear(first,item);
103     break;
104 case 4:
105     display(first);
106     break;
107 default: return 0;
108 }
109 }
110 }
```

1.Insert front  
2.Insert At:  
3.Insert rear  
4.Display

1

Enter element to be inserted

10

1.Insert front  
2.Insert At:  
3.Insert rear  
4.Display

1

Enter element to be inserted

20

1.Insert front  
2.Insert At:  
3.Insert rear  
4.Display

3

Enter element to be inserted

30

1.Insert front  
2.Insert At:  
3.Insert rear  
4.Display

4

Elements of the list are :

20

10

30

1.Insert front  
2.Insert At:  
3.Insert rear  
4.Display

2

Enter element to be inserted

40

Enter position after which to enter

2

30

1.Insert front  
2.Insert At:  
3.Insert rear



4.Display

4

Elements of the list are :

20

10

40

30

1.Insert front

2.Insert At:

3.Insert rear

4.Display