

→ DS Lab Program 5

```
#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 = first;
```

```
    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;
}

void insert_at (NODE first, int item) {
    printf ("Enter position after which to enter:");
    int pos;
    scanf ("%d", &pos);
    NODE bef = getnode ();
    bef = first;
    NODE nrt = getnode ();
    NODE temp = getnode ();
    temp -> info = item;
    temp -> link = NULL;
    for (int i = 1; i < pos; i++)
        bef = bef -> link;
    nrt = bef -> link;
    bef -> link = temp;
    temp -> link = nrt;
}

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

```



```

int main() {
    int item, ch;
    NODE first = NULL;
    for (;;) {
        printf("\n 1. Insert front\n 2. Insert At\n 3.
            Insert rear\n 4. Display\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 insert\n");
                    scanf("%d", &item);
                    insert_at(first, item);
                    break;
            case 3: printf("Enter element to insert\n");
                    scanf("%d", &item);
                    first = insert_rear(first, item);
                    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)

```



```

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 }
65
66 void display(NODE first)
67 {
68     if(first==NULL)
69     {
70         printf("List is empty\n");
71         return;
72     }
73     printf("Elements of the list are : \n");
74     for(NODE i=first;i!=NULL;i=i->link)
75         printf("%d\n",i->info);
76 }
77
78 int main()
79 {
80     int item,ch;
81     NODE first=NULL;
82     for(;;)
83     {
84         printf("\n1.Insert front\n2.Insert At:\n3.Insert rear\n4.Display\n");
85         scanf("%d",&ch);

```

```

78  int main()
79  {
80      int item,ch;
81      NODE first=NULL;
82      for(;;)
83      {
84          printf("\n1.Insert front\n2.Insert At:\n3.Insert rear\n4.Display\n");
85          scanf("%d",&ch);
86          switch(ch)
87          {
88              case 1:
89                  printf("Enter element to be inserted\n");
90                  scanf("%d",&item);
91                  first = insert_front(first,item);
92                  break;
93              case 2:
94                  printf("Enter element to be inserted\n");
95                  scanf("%d",&item);
96                  insert_at(first,item);
97                  break;
98              case 3:
99                  printf("Enter element to be inserted\n");
100                 scanf("%d",&item);
101                 first = insert_rear(first,item);
102                 break;
103             case 4:
104                 display(first);
105                 break;
106             default: return 0;
107         }
108     }
109 }
110

```

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
1
Enter element to be inserted
10

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
2
Enter element to be inserted
25
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 :
10
20
25
30

1.Insert front
2.Insert At:
3.Insert rear
4.Display
7