

Lab-Program-18

```
#include <stdio.h>
#include <malloc.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);
}
```



```
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)
    {
        return temp;
    }
    temp->link = first;
    first = temp;
    return first;
}
```


NODE delete_front(NODE first)

```
{  
    NODE temp;  
    if (first == NULL)  
    {  
        printf("List is Empty & cannot delete\n");  
        return first;  
    }  
    temp = first;  
    temp = temp->link;  
    printf("Item Deleted at Front end is = %d\n", first->info);  
    free(first);  
    return temp;  
}
```

NODE insert_rear(NODE first, int item)

```
{  
    NODE temp, cur;  
    temp = getnode();  
    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 delete_rear(NODE first)
```

```
{
```

```
    NODE cur, Prev;
```

```
    if (first == NULL)
```

```
{
```

```
        Printf("List is empty cannot delete\n");
```

```
        return first;
```

```
}
```

```
    if (first -> link == NULL)
```

```
{
```

```
        Printf("Item deleted is %d\n", first -> info);
```

```
        free (first);
```

```
        return NULL;
```

```
}
```

```
    Prev = NULL;
```

```
    cur = first;
```

```
    while (cur -> link != NULL)
```

```
{
```

```
        Prev = cur;
```

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

```
}
```



```
Printf("Item Deleted at Rear and is  
= %d\n", cur->info);
```

```
free(cur);
```

```
Prev->link = NULL;
```

```
return first;
```

```
}
```

```
void display (NODE first)
```

```
{
```

```
    NODE temp;
```

```
    if (first == NULL)
```

```
        Printf("List is Empty cannot display items\n");
```

```
        for (temp = first; temp != NULL; temp = temp->link)
```

```
        {
```

```
            Printf("%d\n", temp->info);
```

```
        }
```

```
    }
```

```
int main()
```

```
{
```

```
    int item, choice;
```

```
    NODE first = NULL;
```

```
    for(;;)
```

```
    {
```

```
        Printf("1: Insert - Front\n 2: Delete - Front\n 3: Insert - Rear\n 4: Delete - Rear\n 5: Display - List\n 6: Exit\n");
```

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


switch (choice)

{

case 1: Printf("Enter the item at Front-end\n");
scanf("%d", &item);
first = insert_front(first, item);
break;

case 2: first = delete_front(first);
break;

case 3: Printf("Enter the item at Rear-end\n");
scanf("%d", &item);
first = insert_rear(first, item);
break;

case 4: first = delete_rear(first);
break;

case 5: display(first);
break;

default: exit(0);
break;

}

}

return 0;

}

```

1  #include<stdio.h>
2  #include<malloc.h>
3  #include<process.h>
4  struct node
5  {
6      int info;
7      struct node *link;
8  };
9  typedef struct node *NODE;
10 NODE getnode()
11 {
12     NODE x;
13     x=(NODE)malloc(sizeof(struct node));
14     if(x==NULL)
15     {
16         printf("Memory is Full\n");
17         exit(0);
18     }
19     return x;
20 }
21 void freenode(NODE x)
22 {
23     free(x);
24 }
25 NODE insert_front(NODE first,int item)
26 {
27     NODE temp;
28     temp=getnode();
29     temp->info=item;
30     temp->link=NULL;
31     if(first==NULL)
32     {
33         return temp;
34     }
35     temp->link=first;
36     first=temp;
37     return first;
38 }

```

```

36     first=temp;
37     return first;
38 }
39 NODE delete_front(NODE first)
40 {
41     NODE temp;
42     if(first==NULL)
43     {
44         printf("List is Empty cannot delete\n");
45         return first;
46     }
47     temp=first;
48     temp=temp->link;
49     printf("Item Deleted at Front end is = %d\n",first->info);
50     free(first);
51     return temp;
52 }
53 NODE insert_rear(NODE first,int item)
54 {
55     NODE temp,cur;
56     temp=getnode();
57     temp->info=item;
58     temp->link=NULL;
59     if(first==NULL)
60         return temp;
61     cur=first;
62     while(cur->link!=NULL)
63         cur=cur->link;
64     cur->link=temp;
65     return first;
66 }
67 NODE delete_rear(NODE first)
68 {
69     NODE cur,prev;
70     if(first==NULL)
71     {
72         printf("List is Empty cannot delete\n");
73         return first;

```



```

73     return first;
74 }
75 if(first->link==NULL)
76 {
77     printf("Item Deleted is %d\n",first->info);
78     free(first);
79     return NULL;
80 }
81 prev=NULL;
82 cur=first;
83 while(cur->link!=NULL)
84 {
85     prev=cur;
86     cur=cur->link;
87 }
88 printf("Item Deleted at Rear end is = %d\n",cur->info);
89 free(cur);
90 prev->link=NULL;
91 return first;
92 }
93 void display(NODE first)
94 {
95     NODE temp;
96     if(first==NULL)
97         printf("List is Empty cannot display items\n");
98     for(temp=first;temp!=NULL;temp=temp->link)
99     {
100         printf("%d\n",temp->info);
101     }
102 }
103 int main()
104 {
105     int item,choice;
106     NODE first=NULL;
107     for(;;)
108     {
109         printf("1:Insert_Front\n2>Delete_Front\n3:Insert_Rear\n4>Delete_Rear\n5:Display_List\n6:Exit\n");
110         printf("Enter the Choice :");

```



```

99     {
100         printf("%d\n", temp->info);
101     }
102 }
103 int main()
104 {
105     int item, choice;
106     NODE first=NULL;
107     for(;;)
108     {
109         printf("1:Insert_Front\n2:Delete_Front\n3:Insert_Rear\n4:Delete_Rear\n5:Display_List\n6:Exit\n");
110         printf("Enter the Choice :");
111         scanf("%d", &choice);
112         switch(choice)
113         {
114             case 1:printf("Enter the Item at Front-end\n");
115                     scanf("%d", &item);
116                     first=insert_front(first, item);
117                     break;
118             case 2:first=delete_front(first);
119                     break;
120             case 3:printf("Enter the Item at Rear-end\n");
121                     scanf("%d", &item);
122                     first=insert_rear(first, item);
123                     break;
124             case 4:first=delete_rear(first);
125                     break;
126             case 5:display(first);
127                     break;
128             default:exit(0);
129                     break;
130         }
131     }
132     return 0;
133 }
134
135
136

```



```
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
List is Empty cannot display items
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :1
Enter the Item at Front-end
1
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :1
Enter the Item at Front-end
2
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :1
Enter the Item at Front-end
3
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
```


5:Display_List

6:Exit

Enter the Choice :5

3

2

1

1:Insert_Front

2:Delete_Front

3:Insert_Rear

4:Delete_Rear

5:Display_List

6:Exit

Enter the Choice :3

Enter the Item at Rear-end

5

1:Insert_Front

2:Delete_Front

3:Insert_Rear

4:Delete_Rear

5:Display_List

6:Exit

Enter the Choice :3

Enter the Item at Rear-end

10

1:Insert_Front

2:Delete_Front

3:Insert_Rear

4:Delete_Rear

5:Display_List

6:Exit

Enter the Choice :3

Enter the Item at Rear-end

15

1:Insert_Front

2:Delete_Front

3:Insert_Rear

4:Delete_Rear

5:Display_List

6:Exit

Enter the Choice :5

3

2

Enter the Choice :5

3
2
1
5
10
15
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit

Enter the Choice :2

Item Deleted at Front end is = 3

1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit

Enter the Choice :2

Item Deleted at Front end is = 2

1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit

Enter the Choice :2

Item Deleted at Front end is = 1

1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit

Enter the Choice :4

Item Deleted at Rear end is = 15

1:Insert_Front
2:Delete_Front
3:Insert_Rear


```
6:Exit
Enter the Choice :4
Item Deleted at Rear end is = 15
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
Item Deleted at Rear end is = 10
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
Item Deleted is 5
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :2
List is Empty cannot delete
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :4
List is Empty cannot delete
1:Insert_Front
2:Delete_Front
3:Insert_Rear
4:Delete_Rear
5:Display_List
6:Exit
Enter the Choice :5
```

5:Display_List

6:Exit

Enter the Choice :4

Item Deleted is 5

1:Insert_Front

2>Delete_Front

3:Insert_Rear

4>Delete_Rear

5:Display_List

6:Exit

Enter the Choice :2

List is Empty cannot delete

1:Insert_Front

2>Delete_Front

3:Insert_Rear

4>Delete_Rear

5:Display_List

6:Exit

Enter the Choice :4

List is Empty cannot delete

1:Insert_Front

2>Delete_Front

3:Insert_Rear

4>Delete_Rear

5:Display_List

6:Exit

Enter the Choice :5

List is Empty cannot display items

1:Insert_Front

2>Delete_Front

3:Insert_Rear

4>Delete_Rear

5:Display_List

6:Exit

Enter the Choice :6

(program exited with code: 0)

Press any key to continue . . .