

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

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *link;
```

```
};
```

```
typedef struct node *NODE;
```

```
NODE getnode()
```

```
{
```

```
    NODE x;
```

```
    x = malloc(sizeof(NODE));
```

```
    if (x == NULL)
```

```
{
```

```
        printf("Memory Full !!!\n");
```

```
        exit(0);
```

```
    }
```

```
    return x;
```

```
}
```

```
void freenode(NODE x)
```

```
{
```

```
    free(x);
```

```
}
```

```
NODE insert-rear (NODE first, int item)
```

```
NODE temp, cur;
```

```
temp = getnode();
```

```
temp->info = item;
```

```
temp->link = NULL;
```

```
cur = first;
```

if (first == NULL)

return temp;

while (cur->link != NULL)

{

cur = cur->link

}

cur->link = temp

return first;

}

NODE delete\_front (NODE first)

{

NODE temp;

if (first == NULL)

{

printf ("Queue empty\n");

return first;

}

temp = first;

temp = temp->link;

printf ("Item deleted at front end is = %d\n",

first->info);

free (first);

return temp;

}

void display (NODE first)

{

NODE temp;

if (first == NULL)

{

printf ("List is empty\n");

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

{



```
printf("%d\n", temp → info);
```

```
}
```

```
int main()
```

```
{
```

```
int item, choice, pos;
```

```
NODE first = NULL;
```

```
for(;;)
```

```
{
```

```
printf("\n 1.Insert-rear\n 2.Delete-rear\n 3.Display\n 4.Exit\n");
```

```
printf("Enter your choice\n");
```

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

```
switch(choice)
```

```
{
```

```
case 1: printf("Enter item at rear end\n");
```

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

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

```
break;
```

```
case 2: first = delete-front(first);
```

```
break;
```

```
case 3: display(first);
```

```
break;
```

```
default: exit(0);
```

```
break;
```

```
}
```

```
}
```

```
}
```

```

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     NODE x;
10    x=(NODE)malloc(sizeof(struct node));
11    if(x==NULL) {
12        printf("mem full\n");
13        exit(0);
14    }
15    return x;
16 }
17 void freenode(NODE x) {
18     free(x);
19 }
20 NODE insert_rear(NODE first,int item) {
21     NODE temp,cur;
22     temp=getnode();
23     temp->info=item;
24     temp->link=NULL;
25     if(first==NULL)
26         return temp;
27     cur=first;
28     while(cur->link!=NULL)
29         cur=cur->link;
30     cur->link=temp;
31     return first;
32 }
33 NODE delete_front(NODE first) {
34     NODE temp;
35     if(first==NULL) {
36         printf("Queue is empty cannot delete\n");
37         return first;
38     }
39     temp=first;
40     temp=temp->link;
41     printf("item deleted at front-end
42 is=%d\n",first->info);
43     free(first);
44     return temp;
45 }
46 void display(NODE first) {
47     NODE temp;
48     if(first==NULL)
49         printf("Queue empty cannot display
50 items\n");
51     for(temp=first;temp!=NULL;temp=temp->link)
52     {
53         printf("%d\n",temp->info);
54     }
55 }

```

```

51     }
52     }
53 int main() {
54     int item,choice;
55     NODE first=NULL;
56     for(;;) {
57
58         printf("\n1:Insert_rear\n2:Delete_front\n3:Di
59         splay_list\n4:Exit\n");
60         printf("enter the choice\n");
61         scanf("%d",&choice);
62         switch(choice) {
63             case 1:printf("enter the item at
64             rear-end\n");
65                 scanf("%d",&item);
66                 first=insert_rear(first,item);
67                 break;
68             case 2:first=delete_front(first);
69                 break;
70             case 3:display(first);
71                 break;
72             default:exit(0);
73         }
74     }
75 }

```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
10
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
20
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
30
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
10
20
30
```

```
1:Insert_rear
```



```
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=20
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
Queue is empty cannot delete
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
```