```
#include <stdio.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_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;
        }
    }
}
```

```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(struct node));
 if(x==NULL) {
  printf("mem 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;
 if(first==NULL)
  return temp;
 cur=first;
 while(cur->link!=NULL)
  cur=cur->link;
 cur->link=temp;
 return first;
 }
NODE delete_front(NODE first) {
 NODE temp;
 if(first==NULL) {
  printf("Queue 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;
 }
void display(NODE first) {
 NODE temp;
 if(first==NULL)
  printf("Queue empty cannot display
items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
{
  printf("%d\n",temp->info);
```

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

```
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
```