28/10/2020 LAB-4-PROGRAM

1) WAP to simulate the working of a circular queue of integers using an array.
Provide the following operations
a) Insert
b) Delete
c) Display
The program should print appropriate messages for queue empty and queue overflow condition.

CODE:-

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

#define QUE_SIZE 5
int item, front=0, rear=-1, q[QUE_SIZE], count=0;
void insertrear()
{
 if (count== QUE_SIZE)
 {
  printf("queue overflow\n");
 return;
 }
 rear= (rear+1)%QUE_SIZE;
  q[rear]=item;
  count++;
}
int deletefront()
{
 if (count==0) return -1;
 item= q[front];
 front = (front+1)% QUE_SIZE;
  count = count-1;
```

```
return

}
void displayQ()
{
 int i, f;
 if (count ==0)
 {
 printf ("queue is empty\n");
  return;
}
f= front;
printf("Contents of queue \n");
 for (i= 1; i<=count; i++)
 {
  printf ("%d\n", q [f]);
  f= (f+1) % QUE_SIZE;
 }
}
int main()
{
 int choice;
 for (;;)
 {
 printf ("\n1: Insertrear \n2: delefront \n3: display \n4: exit \n");
 printf ("enter the choice \n");
 scanf ("%d", &choice);

 switch (choice)
 {
  case1: printf ("enter the item to be inserted \n");
   scanf ("%d", &item);
    Insertrear();
    break;
  case2: item= deletefront();
2        if (item==-1)
1        printf ("queue is empty \n");
         else
         printf (" item deleted = %d\n", item);
          break;
       case 3: displayQ();
       default : exit (0);
       }
      }
     }
```

EXTRA PROGRAMS

1) Double ended queue

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

# define qsize 5

int f=0, r=-1, ch;
int item, q[10];

int isfull()
{
 return (r==qsize-1)? 1:0;
}
int isempty()
{
 return (f>r)?1:0;
}
void insert_rear()
{
 if (is full())
  {
   printf("queue overflow\n");
   return;
  }
  r= r+1;
  q[r]= item;
}
void delete_front()
 {
  if (isempty())
   {
    printf(" queue empty\n");
    return;
   }
```

```
printf ("item deleted is %d\n", q [(f)f++]);
if (f>r)
{
 f=0;
 r=-1;
}
}

void insert_front()
{
 if(f!=0)
 {
  f=f-1;
  q [f]=item;
  return;
 }
 else if ((f==0)&&(r==-1))
 {
  q [++(r)]=item;
  return;
 }
 else
 printf ("insertion not possible \n");
}
 void delete_rear()
  {
   if (isempty())
   {
    printf ("queue is empty\n");
    return;
   }
```

```
printf ("item deleted is %d\n", q [(r)--]);
 if (f>r)
  {
   f=0;
   r=-1;
}
}
void display()
 {
  int i;
  if (isempty())
  {
   printf ("queue empty\n");
    return;
  }
  for ( i= f ; i<=r ; i++)
  printf ("%d\n", q [i]);
 }
void main ()
{
 for (;;)
 {
  printf (" 1. insert_ rear \n2. insert_front \n3. delete_rear\n4.
         delete - fuont \n5. display\n6. exit \n");
  printf ("enter choice \n");
  scanf ("%d", &ch);
  switch (ch)
  {
   case 1: printf ("enter the item\n");
           scanf ("%d", &item);
           insert _ rear();
           break;
case 2: printf ("enter the item\n");
        scanf ("%d", &item);
        insert-front ();
        break;
case 3: delete-rear ();
        break;
case 4: delete-front();
        break;
case 5: display();
        break;
default : return ;
}
}
getch();
}
```

2] Input Restricted program:-

```
#include <stdio.h>
#include < conio.h>
#include <stdlib.h>
# define qsize 3
int f=0, r= -1, ch;
int item, q [10];
int isfull()
{
return (r== qsize-1)? 1:0;
}
int isempty ()
{
return(f>r)? 1:0;
}
```

```
{ if (isfull())
  {
   printf ("queue overflow \n");
   return;
  }
  r= r+1;
  q[r]=item;
  }
  void delete_front()
   {
    if (isempty())
    {
     printf("queue empty \n");
     return;
    }
    printf("item deleted is %d \n", q[(f)++]);
    if (f>r)
     {
      f=0;
      r=-1;
     }
    }
    void delete_rear()
     {
      if (isempty())
      {
       printf ("queue is empty \n");
       return;
      }
```

```
printf ( "Item deleted is %d \n", q
 if (f>r)
 {
  f=0;
  r=-1;
 }
}
void display()
 {
 int i;
 if (isempty())
 {
  printf ("queue empty \n");
  return;
 }
 for (i=f ; i<=r ; i++)
 printf ("%d \n", q[i]);
}
 void main()
 {
  for(;;)
 {
  printf ("1. insert_rear \n2. delete_rear \n3. delete-front \n
        \n6. exit \n");
  printf ("enter the choice \n");
  scanf ("%d", &ch);
  switch(ch)
  {
   case 1: printf ("enter the item \n");
   scanf ("%d", &item);
   insert_rear();
   break;
case 2: delete-rear();
 break;
case 3: delete- front();
 break;
 case 4:display();
 break;
 default : exit(0);
}
}
getch();
}
```

3] Output Restricted Dequeue

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define qsize 3
 int f=0,r=-1,ch;
 int item, q[10];
 int isfull()
  {
   return (r==qsize-1)?1:0;
  }
  int isempty()
  {
   return (f>r)?1:0;
  }
  void insert_rear()
  {
   if (isfull())
```

```
{
 printf ("queue overflow \n");
 return ;
}
r= r+1;
q [r]= item;
}
void delete_front ()
{
 if (isempty ())
 {
  printf ("queue empty \n");
  return;
 }
 printf ("item deleted is %d \n", q[(f)++]);
 if (f>r)
 {
 f=0;
 r=-1;
}
}
void insert_front()
{
 if (f!=0)
 {
  f= f-1;
  q [f]= item;
  return;
 }
 else if ((f==0) && (r==-1))
 {
  q[++(r)]=item;
```

```
return;
}
else
printf ("insertion not possible \n");
}
void display ()
{
int i;
if (empty())
{
printf ("queue empty \n");
return;
}
for (i= f; i<=r; i++)
printf ("%d \n", q[i]);
}
void main()
{
for(;;)
{
printf ("1. insert_rear \n2. insert_front \n3. delete_front
     \n4. display \n6. exit \n");
printf ("enter choice \n");
scanf ("%d", &ch);
switch (ch)
{
 case 1: printf ("enter the item \n");
 scanf ("%d", &item);
 insert_rear();
 break;
case 2: printf ("enter the item\n");
  scanf ("%d", &item);
  insert_front();
  break;
 case 3: delete_front();
  break;
  default : exit(0);
 }
}
getch();
}
```