Assignment :- 6

(1) Circular Queue

(a) Insertion

(b) Deletion

(c) Display.

```
#include <stdlib.h>
#include <stdio.h>
#define max 5
int front=-1, rear=-1;
int cQueue[max];
void insert();
int delete();
void display();
int main()
{
    int w, no;
    for(;;)
```

```
{
printf("\n1. Insert");
printf("\n2. Delete");
printf("\n3. Display");
printf("\n4. Exit");
printf("\n Enter what you want:");
scanf("%d", &w);
switch(w)
{
case 1:
    insert();
    break;
case 2:
    no = delete();
    break;
case 3:
    display();
    break;
case 4:
    exit(1);
default:
    printf("\n Invalid choice!!\n");
    }
}
```

```
Void insert()
{
  int no;
  If ((front == 0 && rear == max-1) || front == rear+1)
  {
    printf ("\n Circular Queue is full !\n");
    return;
  }
  printf("\n Enter a number to Insert: ");
  Scanf ("%d", &no);
  if (front == -1)
    front = front+1;
  if (front = max-1)
    rear=0;
  else rear = rear+1;
    C Queue[rear] = no;
  }
  int delete()
{
    int e;
```

```
if (front == -1)

{ printf ("\n The Circular Queue is Empty !!!\n");

 }

e = cQueue [front];

if (front == max-1)

  front = 0;

else if (front == rear)

 {

   front = -1;

    rear = -1;

}

else front = front+1;

  printf ("\n %d was deleted !!\n", e);

    return e;

 }

 void display()

{

 int i;

 if (front == -1)

{

  printf ("\n The Circular Queue is Empty ! Nothing to Display !!\n");

   } return;
```

```
i=front;
if (front<=rear)
{
  printf("\n\n");
  while (i<=rear)
    printf("%d", Queue[i++]);
    printf("\n");
}
else
{
  printf("\n\n");
  while (i<max-1)
    printf("%d", Queue[i++]);
    i=0;
  while (i<=rear)
    printf("%d", Queue[i++]);
    printf("\n");
  }
}
```

(3)

push operation :-

— If stack is Empty

— push an element init and set min & top point to it

— else

— push element on the top (its next min will be Null)

— If (element inserted is less than element at min)

— top -> next min = min

— min = top;

Pop operation :-

— If top element is also min element

— min = min -> next min

— pop the element at top

(2)

- The only thing that we have to count the number of zeros in each of the numbers and simultaneously delete the zero in that number
- So now we have numbers with no trailing zeros and a count variable with the number of zeros in every number.

- The counting is done by using divide (/) and module (%) with 10 and also using loops (mainly while loop is used).
- finally multiple the numbers with no trailing zeros and using a loop print zeros (as such as the number in count variable) at the end.