**PROGRAM-5**

***Q. Design and implement a given type of queue in C (Ordinary queue/Circular queue) using Array implementation and linked list implementation. Also demonstrate the working with suitable inputs. Display appropriate messages in case of exceptions.***

**“Circular Queue Implementation using Arrays”**

* **Theory:**

A circular queue is an extended variant of a conventional queue where the start and last elements are interconnected. resulting in a structure resembling a circle.

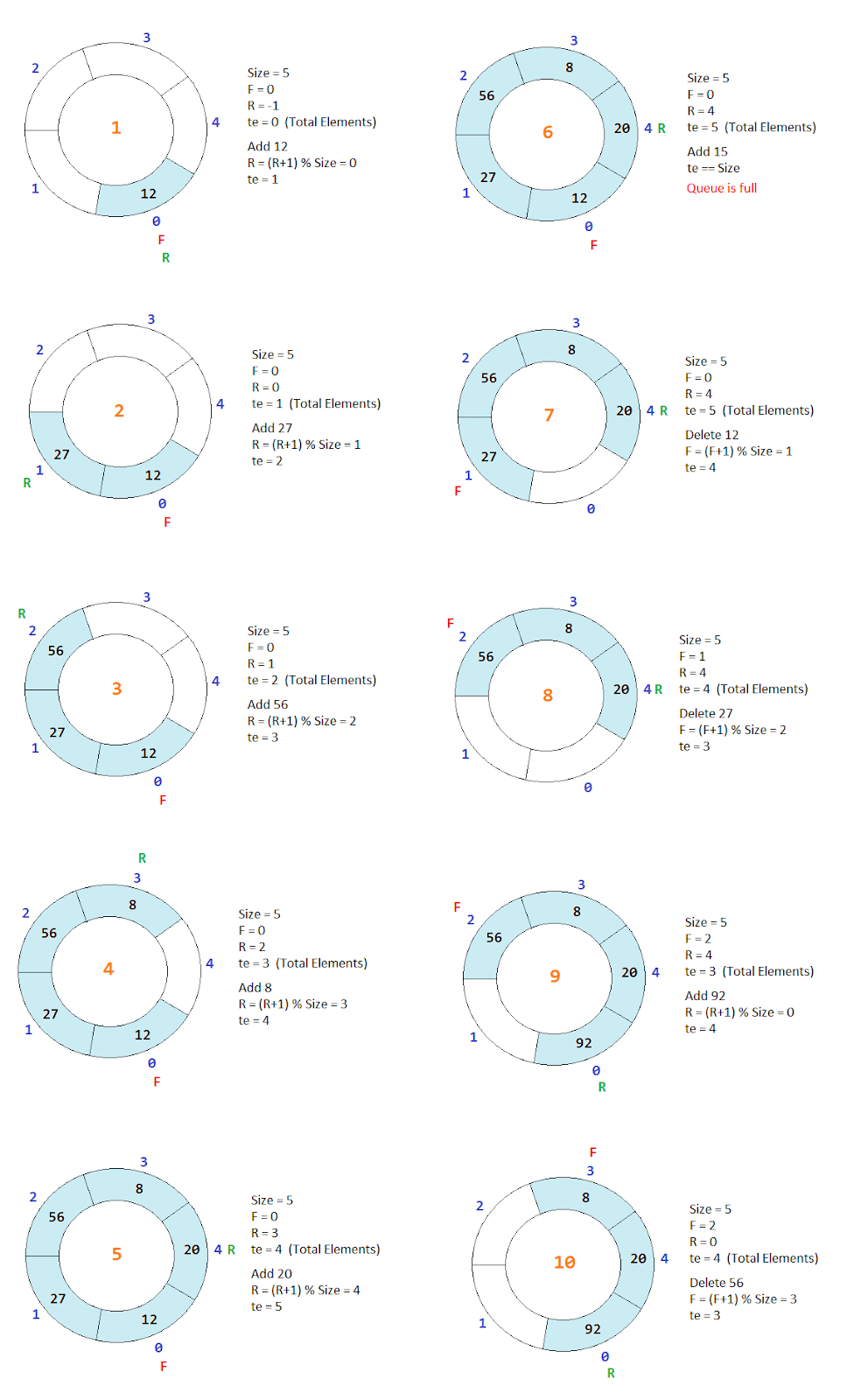
To avoid the dilemma with the array implementation of a queue, we use a circular queue. The issue is that even if there are still spaces in the line when the back of the queue fills up and we try to add a new element after deleting some from the front, it says "Queue is full."

On the circular queue, two operations are performed.

Add - When we add an element in the circular queue.

Delete - When we delete an element from the circular queue.

We can see that adding a new element to the circular queue causes the variable R to be increased by R=(R+1) % Size, the new element to be added at the new position of R, and te to be incremented by 1. The variable F increases by F=(F+1) % Size when an element is removed from the circular queue, while te decreases by 1 when the same happens.



* **Algorithm:**

**Insertion:**

Step 1: IF (REAR+1)%MAX = FRONT

Write " OVERFLOW "

goto step 4

[End OF IF]

Step 2: IF FRONT = -1 and REAR = -1

SET FRONT = REAR = 0

ELSE IF REAR = MAX - 1 and FRONT! = 0

SET REAR = 0

ELSE

SET REAR = (REAR + 1) % MAX

[END OF IF]

Step 3: SET QUEUE[REAR] = VAL

Step 4: EXIT

**Deletion:**

Step 1: IF FRONT = -1

Write " UNDERFLOW "

goto Step 4

[END of IF]

Step 2: SET VAL = QUEUE[FRONT]

Step 3: IF FRONT = REAR

SET FRONT = REAR = -1

ELSE

IF FRONT = MAX -1

SET FRONT = 0

ELSE

SET FRONT = FRONT + 1

[END of IF]

[END OF IF]

Step 4: EXIT

* **Code:**

#include <stdio.h>

#define MAX\_SIZE 5 // a queue of size 5 is defined

int a[MAX\_SIZE]; //an array is declared

int front = -1; //front and rear are set to -1

int rear = -1;

//------------------------------------------------------------

void enqueue (int x) //insert operation

{

if (front == -1 && rear == -1)

{

front = rear = 0; // front and rear are set to zero

}

else if ((rear + 1) % MAX\_SIZE == front) //exception

{

printf("queue is full\n"); //Overflow

return;

}

else

rear = (rear + 1) % MAX\_SIZE; //rear is incremented

a[rear] = x; //value is inserted into the queue

printf("Value inserted\n");

}

//---------------------------------------------------------------

void dequeue() //delete operation

{

int val;

if (front == -1 && rear == -1) // exception

{

printf("queue is empty \n"); // Underflow condition

return;

}

val=a[front]; //item is dequeued i.e, deleted from front end

if (front == rear)

{

front = rear = -1; //if front ==rear both are set to -1

}

else

{

front = (front + 1) % MAX\_SIZE; //front is incremented

}

printf("element deleted is %d\n",val);

//deleted value is printed on the console

}

//------------------------------------------------------------

void display() //display operation

{

int i;

for (i =front; i <=rear; i++) //for loop to print elements

{

printf("%d\n",a[i]);

}

}

//---------------------------------------------------------------

int main() //main function

{

int ch,x; //ch and x are initilaized

while (ch!=4) //loop

{

//main menu is displayed

printf("Main Menu\n");

printf("-----------------\n");

printf("1. Insert an element\n");

printf("2. Delete an element\n");

printf("3. Display the elements\n");

printf("4. Exit\n");

printf("Enter Choice:");

scanf("%d",&ch); //user enters the choice

switch(ch)

{

case 1: printf("Enter the value:");

scanf("%d",&x); //user enters the value

enqueue(x); //enqueue is called with x as parameter

break;

case 2: dequeue();

break;

case 3: display(); //function call

break;

case 4: exit(0); //switch case to perform operations

break;

default:

printf("Wrong Choice");

}

}

}

//------------------------------------------------------

**“SCREENSHOTS OF OUTPUT”**

