Algorithm_Insert_Circular_Queue_using_Array()

Input: A circular queue implemented using an array, say A, an element to be inserted, say ITEM and FRONT, a variable that will hold the index of the item inserted first in the queue and REAR, a variable that will hold the index of the item inserted last in the queue.

Output: ITEM successfully inserted at the REARth position of the circular queue otherwise suitable overflow message.

Data Structure used: An array A[L..U] where L = Lower index of the array, U = Upper index of the array and SIZE = U – L + 1

Steps:

1. Begin
2. If (FRONT == (REAR + 1) % SIZE)
3. Then
4. Print "Queue Overflow, Item can't be inserted “
5. Else
6. Set REAR = (REAR + 1) % SIZE
7. Set A[REAR] = ITEM
8. If (FRONT = L – 1)
9. Then
10. Set FRONT = FRONT + 1
11. End If
12. End If
13. End

Algorithm_Delete_Circular_Queue_using_Array()

Input: A circular ueue implemented using an array, say A, FRONT, a variable that will hold the index of the item inserted first in the queue and REAR, a variable that will hold the index of the item inserted last in the queue.

Output: An item, say ITEM successfully deleted from the FRONTth position of the queue otherwise suitable underflow message.

Data Structure used: An array A[L..U] where L = Lower index of the array, U = Upper index of the array and SIZE = U – L + 1

Steps:

1. Begin
2. If (FRONT == L – 1)
3. Printf "Queue underflow, No item to delete”
4. Else

```
5.      Set ITEM = A[FRONT]
6.      If (FRONT = REAR)
7.      Then
8.          Set FRONT = REAR = L – 1
9.      Else
10.         Set FRONT = (FRONT + 1) % SIZE
11.     End If
12.     Return ITEM
13. End If
14. End
```

Algorithm_Traverse_Circular_Queue_using_Array()

Input: A circular queue implemented using an array, say A, FRONT, a variable that will hold the index of the item inserted first in the queue and REAR, a variable that will hold the index of the item inserted last in the queue.

Output: The elements of the queue are successfully traversed from the FRONT position till the REAR position otherwise suitable underflow message.

Data Structure used: An array A[L..U] where L = Lower index of the array, U = Upper index of the array and SIZE = U – L + 1

Steps:

```
1.  Begin
2.  If (FRONT = L – 1)
3.  Then
4.      Printf “Queue Underflow, No item to traverse”
5.  Else
6.      If (FRONT ≤REAR)
7.      Then
8.          Set i = FRONT
9.          While i ≤ REAR
10.         Begin
11.             Process (A[i])
12.             Set i = i + 1
13.         End While
14.     Else
15.         Set i = FRONT
16.         While i ≤ U
17.         Begin
18.             Process (A[i])
19.             Set i = i + 1
20.         End While
```

```
21.                 Set i = L
22.                 While (i ≤ REAR)
23.                 Begin
24.                         Process (A[i])
25.                         Set i = i + 1
26.                 End While
27.         End If
28. End If
29. End
```