Book

A Simplified Approach to

Data Structures

Prof.(Dr.)Vishal Goyal, Professor, Punjabi University Patiala

Dr. Lalit Goyal, Associate Professor, DAV College, Jalandhar

Mr. Pawan Kumar, Assistant Professor, DAV College, Bhatinda

Shroff Publications and Distributors

Edition 2014

QUEUE

CONTENTS INCLUDED

- Definition of Queue
  - Introduction of queue
  - Application of queue
- Operations on queue
  - Insertion of element
  - Deletion of element
- Memory representation of Queue
  - Array representation of Queue
  - Linked representation of Queue

INTRODUCTION OF QUEUE

- ***Queue*** is a linear data structure.
- ***Queue*** has two ends Front and Rear.
- Element can be added at ***Rear*** of the queue and the element can be removed from the ***Front*** end of the queue.
- The elements of a queue are processed in the same order as they were added into the queue.

INTRODUCTION OF QUEUE(CONT…)

- Queues are also known as ***FIFO (First In Order First Out)*** list or ***FCFS (First Come First Serve basis) list*** .

- ***Queue*** contrasts with ***STACKS***,which are ***last in first out(lifo).***

Example:

- Queues occur in real life a lot:
  1. Queues at checkout
  2. Queues in banks
- In software systems:
  1. Queue of requests at a web servers

People waiting in queue

APPLICATIONS OF QUEUE

1. Direct applications:
   - Waiting lists, bureaucracy
   - Access to shared resources (e.g., printer)
   - Multiprogramming
2. Indirect applications:
   - Auxiliary data structure for algorithms
   - Component of other data structures

OPERATIONS ON QUEUE

There are two operations on the queue:-

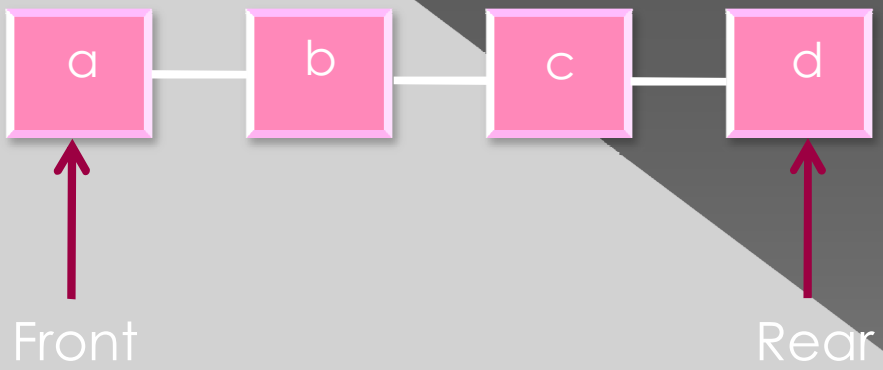
1. Insertion
2. Deletion

INSERTION OPERATION

- ***Insertion*** operation refers to ***addition*** of element in Queue.
- ***Insertion*** operation processed only when there is space in Queue, otherwise it gives ***overflow*** ,it indicates to the user there is no space in queue.
- ***Insertion*** of element in Queue is done from the ***Rear*** end.
- After ***N insertions*** ,the ***Rear*** element of the Queue will occupy ***QUEUE[N]*** or in other words; eventually the ***Queue*** will occupy the last part of the array.

INSERTION OPERATION(Cont…)

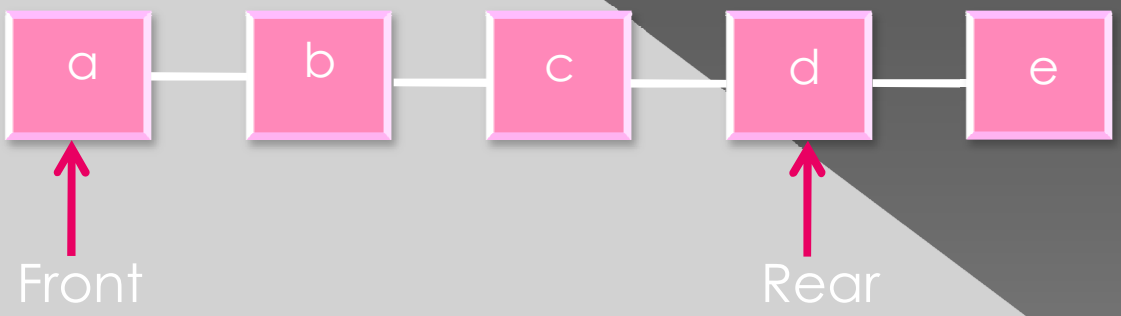
Consider a list of four elements **(a, b, c, d)** where **a** is the front element and **d** is rear element.



A queue with Four Elements

INSERTION OPERATION(Cont…)

New element **e** will be inserted at the rear end, here, after the element **d** as shown in figure below:-

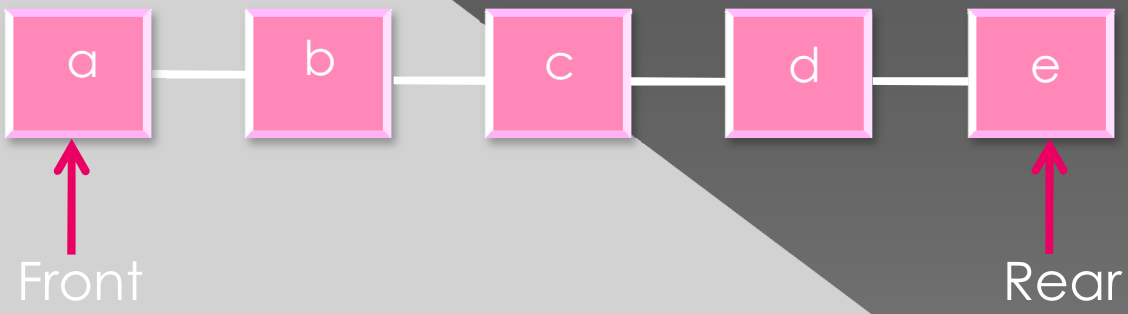


DELETION OPERATION

- ***Deletion*** operation refers to removal of an element from the queue.
- ***Deletion*** operation is processed only when there is element present in the Queue,otherwise it gives ***underflow***,which tells the user that there is no element present in Queue.
- The ***Deletion*** of element is done from the front of Queue.

DELETION OPERATION(Cont…)

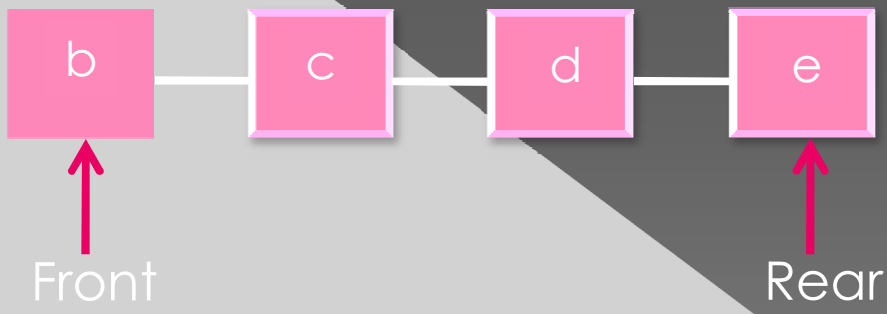
Only element at the front end can be deleted from the queue. Here, the element **a** will be deleted from the queue as shown:



Deleting an element from the queue

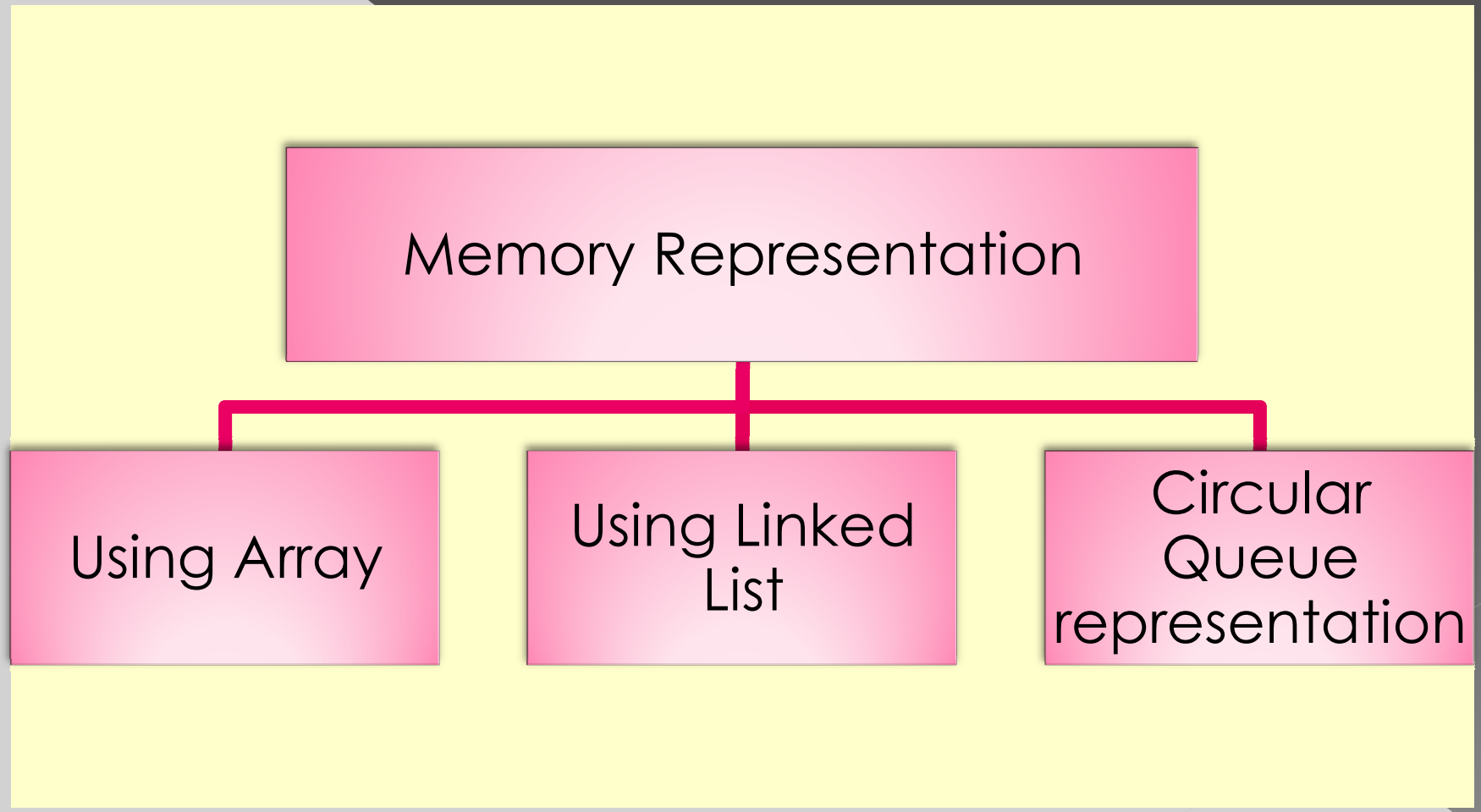
DELETION OPERATION(Cont…)

Another element that can be deleted from the queue is **b** as shown below:



Deleting another element from the queue

MEMORY REPRESENTATION OF QUEUE



ARRAY REPRESENTATION OF QUEUE

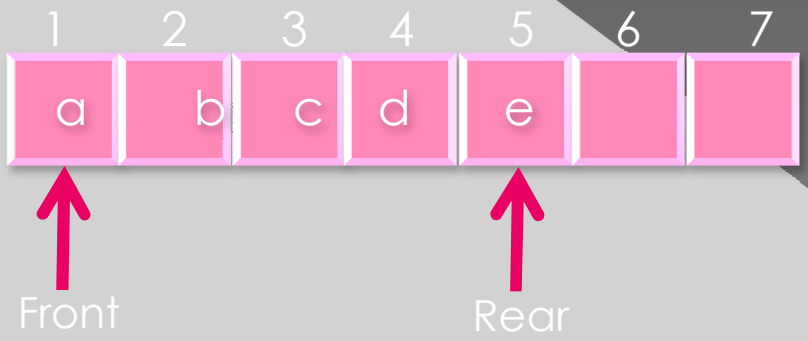
- The elements of the Queue must be of same type (homogenous).
- Maximum size of the queue must be defined before implementing it as array is static ***data structure***.
- Queue grows and shrinks over time but an array has constant size.
- ***First In First Out (FIFO)*** order must be maintained using two variables ***Front*** and ***Rear***

ARRAY REPRESENTATION OF QUEUE (Cont..)

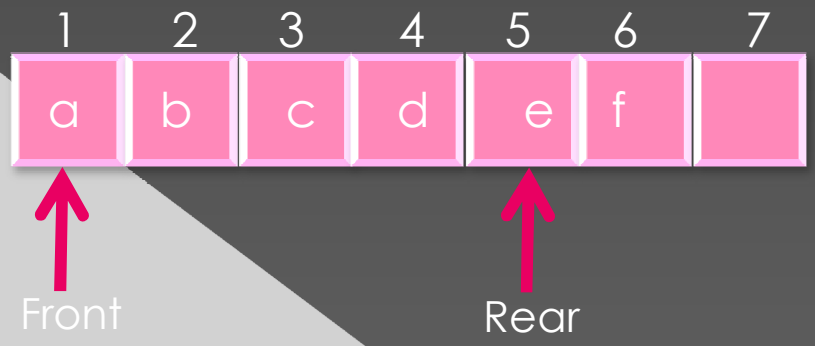
INSERTION:

INSERTION means the addition of element in Queue.Whenever an element is added to the Queue,the value of ***REAR*** is increased by 1;this can be implemented by the assignment

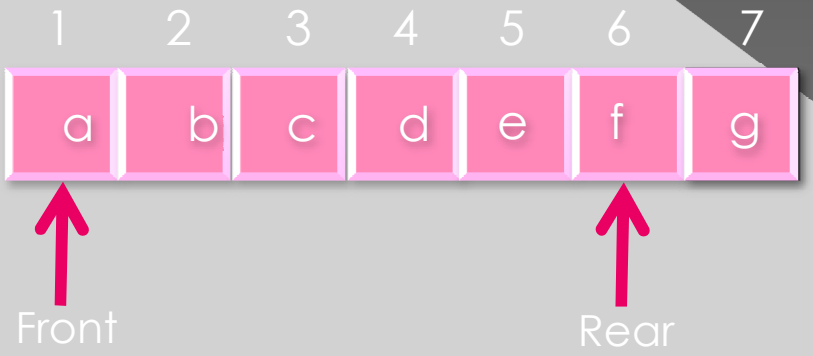
REAR=RAER+1



Queue having 5 elements



Inserting an element f at index 6 in the queue



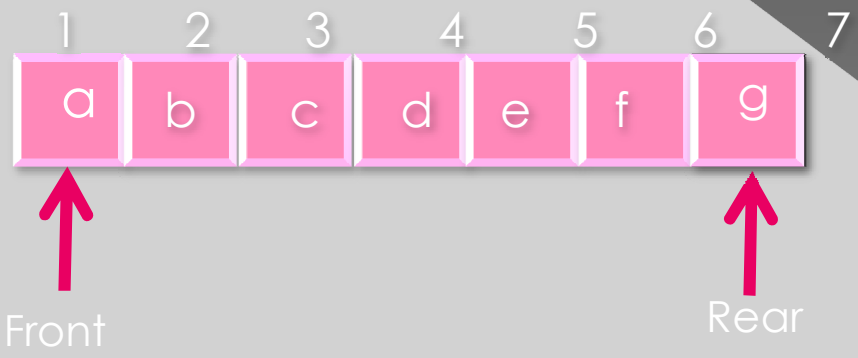
Inserting an element g at index 7 in the queue

ARRAY REPRESENTATION OF QUEUE(Cont…)

DELETION:

The only element at the front of the Queue can be removed and variable ***Front*** of the queue will be incremented by one.After deletion of element value of FRONT is increased by 1;This can be implemented by the assignment

REAR=REAR+1



Deletion of an element from the queue

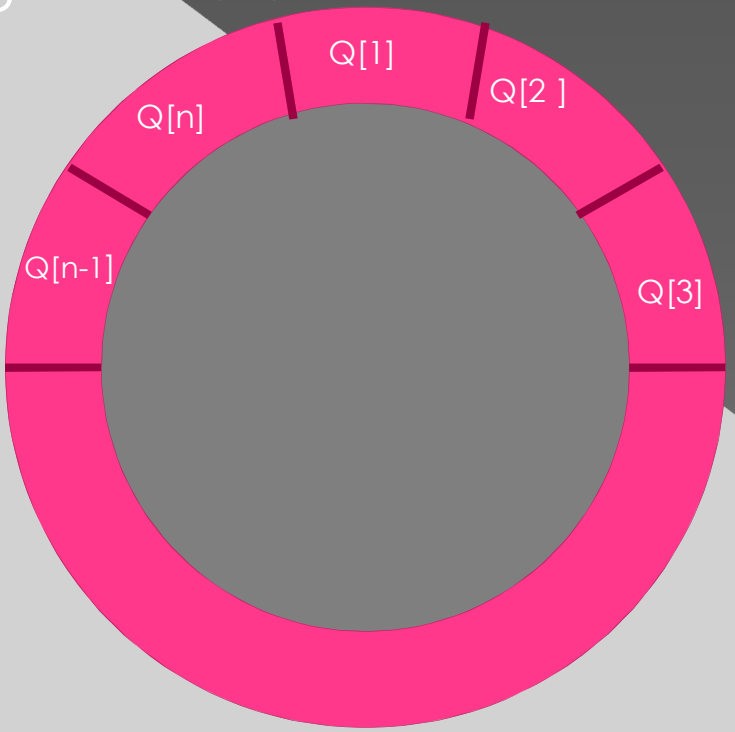
CIRCULAR REPRESENTATION OF QUEUE

In the above mentioned queue, the front positions start vacating during the deletion process. To make full use of space, two cases arise,

- ***Queue*** contrasts with ***STACKS***,which are ***last in first out(lifo).***Shift all the elements in the left after each deletion position.
- Use circular array to implement queue termed as ***circular Queue***.
- Shifting elements in the front positions is not efficient in terms of time, so the circular queue is very efficient option.

CIRCULAR QUEUE

- An array in the form of circle is used.
- After the last index, there it the turn of first index making it circular.



A CIRCULAR ARRAY OF SIZE *n*

OPERATIONS ON CIRCULAR QUEUE

1. *INSERTION*
2. *DELETION*

INSERTION IN CIRCULAR QUEUE

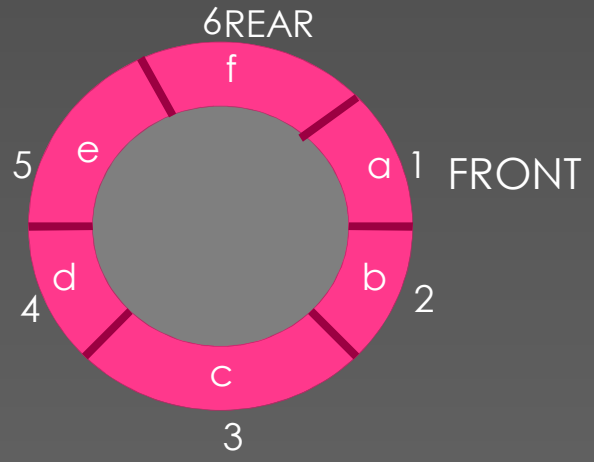
- Before inserting an element, the ***overflow*** condition must be checked.
- If last indexed position is occupied, element will be inserted at the ***first index***.

ALGORITHM INSERTION IN CIRCULAR QUEUE(Cont…)

Insertion of an element 'Data' into the circular queue. The size of the Queue is 'n' i.e. 'n' number of elements can be accommodated in the Queue. Here, lower index is taken as '1' and upper index is taken as 'n'.

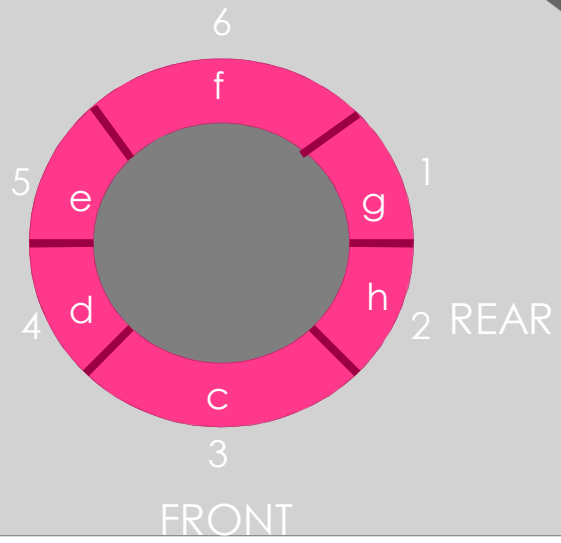
Step 1: If ***FRONT* = 1** and **REAR = n** Then
Print "Queue is full, Overflow Condition"
Exit
[End If]

Step 2 : If **FRONT = REAR+ 1** Then
Print "Queue is full, Overflow Condition"
Exit
[End If]



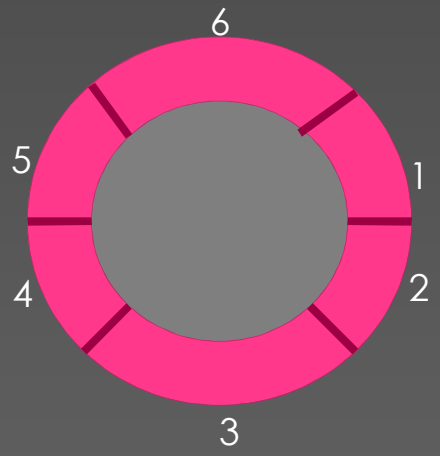
FRONT=1 REAR=6

FRONT=REAR+1



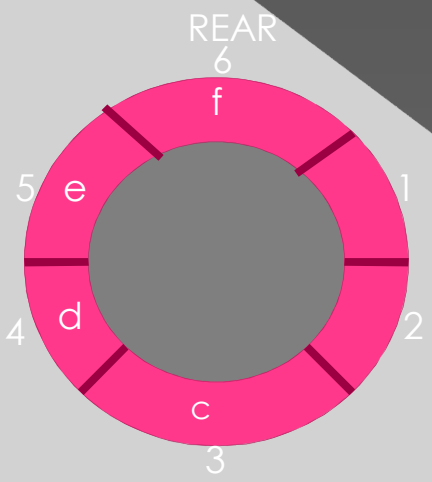
ALGORITHM INSERTION IN CIRCULAR QUEUE(Cont…)

Step 3: If **REAR = NULL** Then
Set **FRONT = 1** and **REAR = 1**



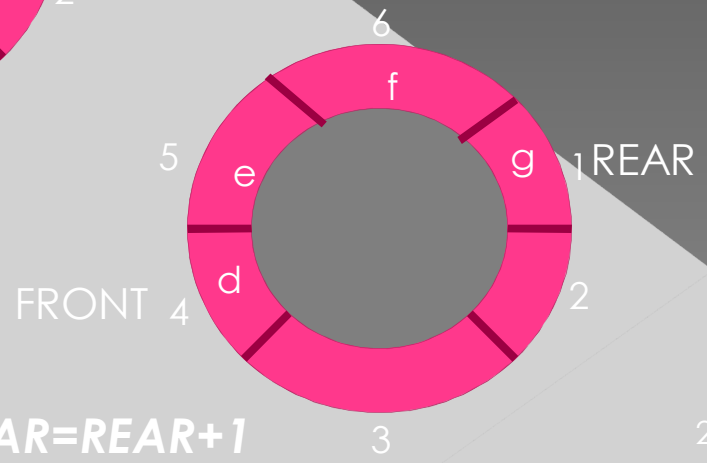
FRONT=0 REAR=0

If **REAR = n** Then
Set **REAR = 1**



REAR=1

Else
Set **REAR = REAR + 1**
[End If]
Step 4: Set **Q[REAR] = DATA**
Step 5: Exit



REAR=REAR+1

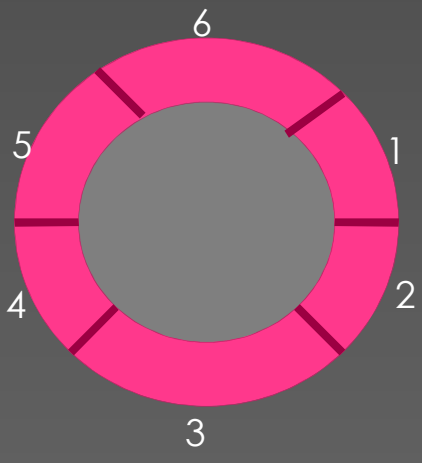
DELETION IN CIRCULAR QUEUE

- Before deleting an element, the **underflow** condition must be checked.
- If **Front** is reached at last index, after deletion **Front** will refer to the first index.

ALGORITHM OF DELETION IN CIRCULAR QUEUE

Deleting an Element from the Queue. The size of the Queue is 'n' i.e. 'n' number of elements can be accommodated in the Queue. Here, lower index is taken as '1' and upper index is taken as 'n'.

Step 1: If **FRONT = NULL** Then
Print: "Queue is empty, Underflow Condition"
Exit
[End If

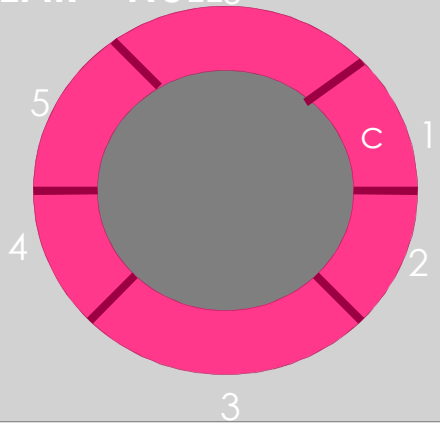


FRONT=0 REAR=0

Step 2: Set **DATA = Q[FRONT]**

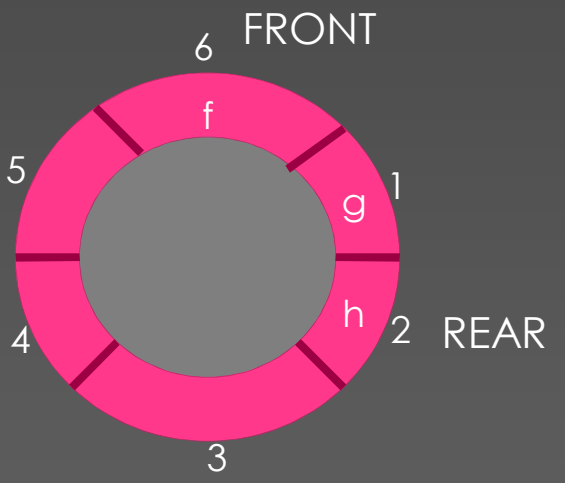
Step 3: If **FRONT = REAR** Then
Set **FRONT = NULL** and **REAR = NULL**

FRONT=REAR



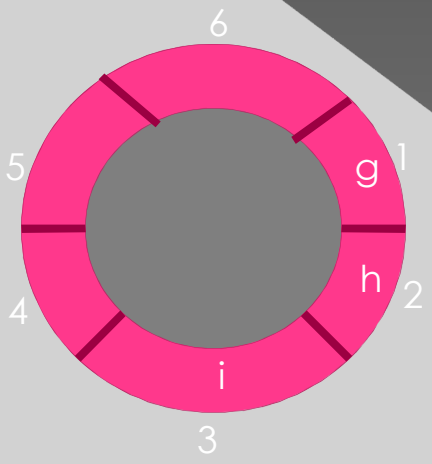
ALGORITHM OF DELETION IN CIRCULAR QUEUE(Cont….)

Else If **FRONT = n** Then
Set **FRONT = 1**



FRONT=6

Else
Set **FRONT = FRONT+ 1**
[End If]

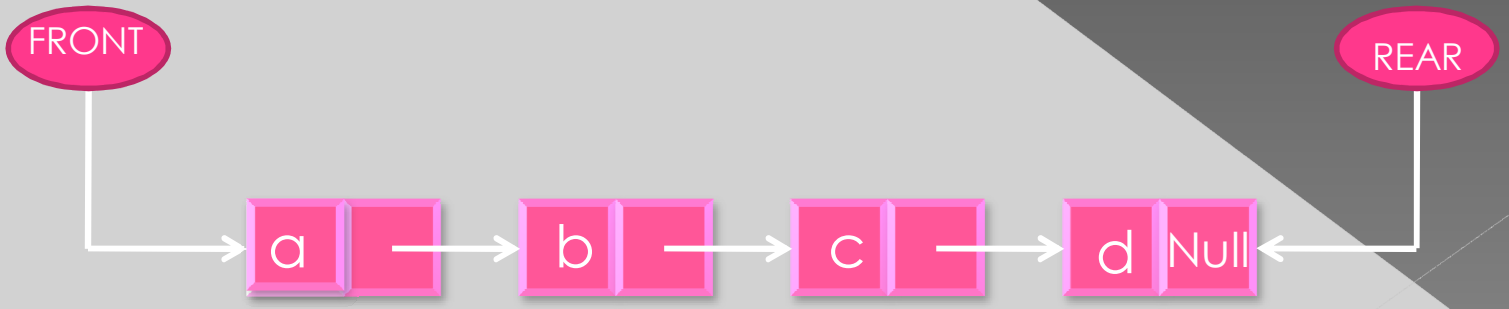


FRONT=1

Step 4: Exit

LINKED LIST REPRESENTATION OF QUEUE

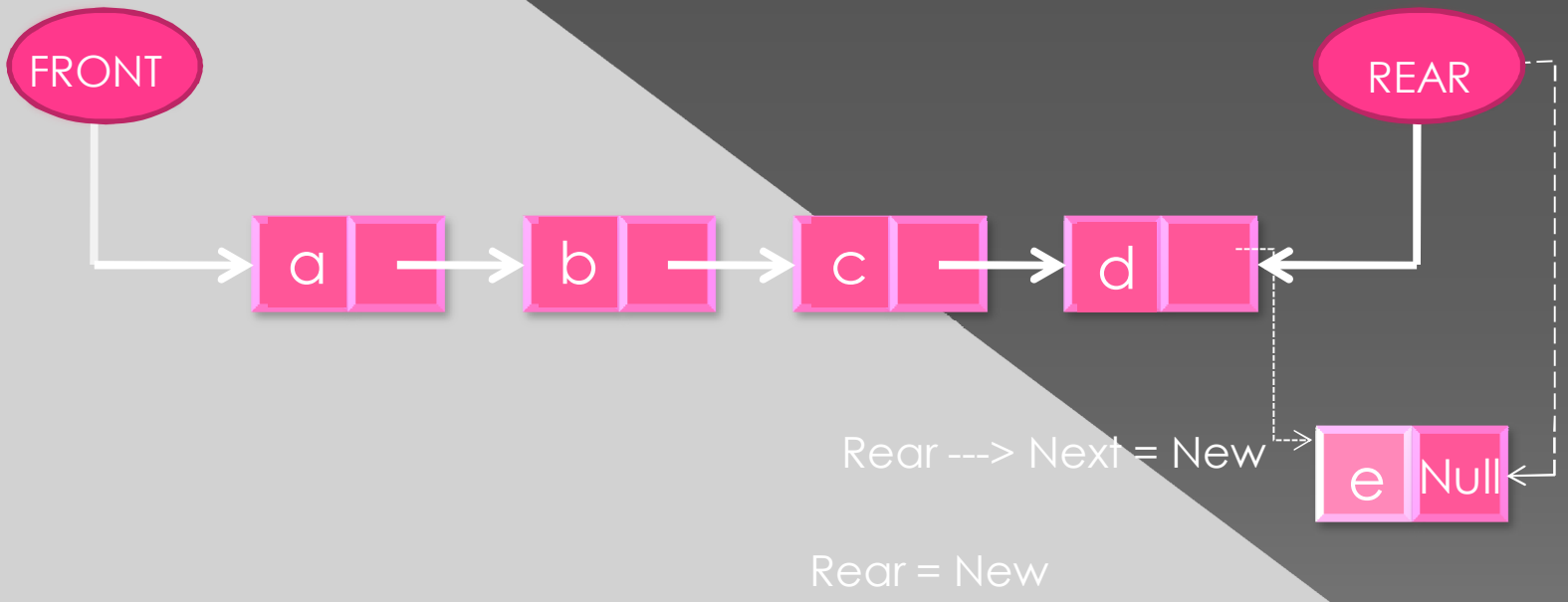
- The elements of the queue may be of different type (hetrogenous).
- size of the queue may be changed at run time (Dynamic data structure).
- ***First In First Out (FIFO)*** order must be maintained using two pointer variables ***Front*** and ***Rear***.
- Holds the address of the first node and the Rear holds the address of the last node of the linked list.



A Queue Maintained using a Linked List

INSERTION IN QUEUE USING LINKED LIST

The ***insertion*** of a new element **e** in the above shown queue can be shown as in figure below:



This insertion of an element 'e' in the queue

ALGORITHM OF INSERTION IN QUEUE USING LINKED LIST

This algorithm inserts a given element 'Data' in a queue which is implemented using a linked list 'Q' having variable FRONT which contains the address of 1st element of the queue and variable Rear which contains the address of last element of the queue.

Step 1: If **FREE = NULL** Then
Print: "No Free Space Available for Insertion"
Exit
[End If]

Step 2: Allocate memory to node **NEW**
Set **NEW = FREE** and **FREE** = **FREE ->NEXT**

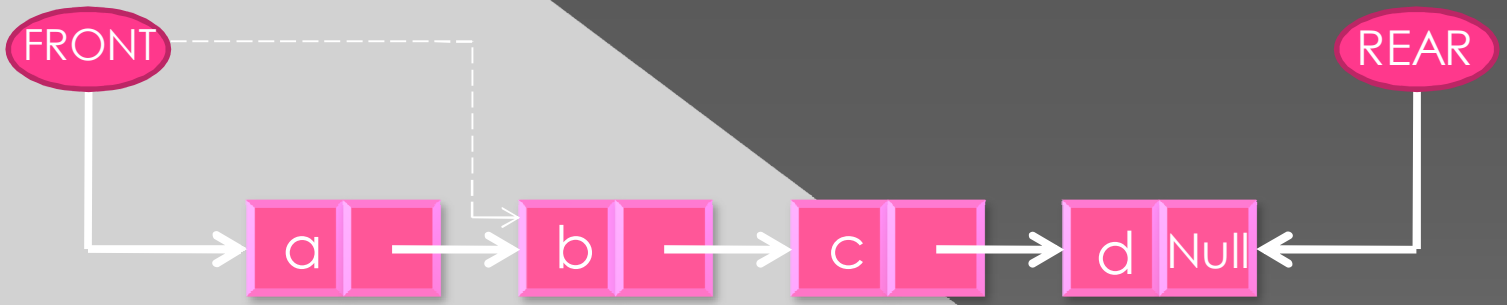
Step 3: Set **NEW-> INFO** = **DATA** and **NEW ->NEXT** = **NULL**

Step 4: If **REAR** = **NULL** Then
Set **FRONT** = **NEW** and **REAR** = **NEW**
Else
Set **REAR ->NEXT** = **NEW** and **REAR** = **NEW**
[End If]

Step 5: Exit

DELETION IN QUEUE USING LINKED LIST

- Deletion of node pointed by Front variable can be done.
- After deletion, Front will point to 2nd node.



Front = Front -> Next

Deletion of an element from the Queue

ALGORITHM OF DELETION IN QUEUE USING LINKED LIST

This algorithm removes an element from a queue which is maintained using linked list 'Q' having variable Front which contains the address of 1st element of the queue and variable Rear which contains the address of least element of the queue.

Step 1: If **FRONT** = **NULL** Then
Print " Queue is Empty"
Exit
[End If]

Step 2: Set **DATA** = **FRONT-> INFO**, **TEMP** = **FRONT**

Step 3: If **FRONT** = **REAR** Then
Set **FRONT** = **NULL** and **REAR** = **NULL**
Else
Set **FRONT**= **FRONT->** Next
[End If]

Step 5: Deallocate memory taken by node **TEMP**
Set **TEMP-> NEXT** = **FREE**, **FREE**= **TEMP**

Step 6: Exit