K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

Data Structure

ENCS205

School of Engineering & Technology (SOET)
K.R. MANGALAM University

UNIT-2
Session 29: Circular Queue & Deque

Recap

Introduction of Queues: Queues are a fundamental data structure used in computer science to manage data in a sequential order. They follow the First In, First Out (FIFO) principle, meaning the first element added to the queue is the first one to be removed. Just like waiting in line at a grocery store, the person who arrives first is served first.

Recap

Operations of Queues

- Enqueue (Insertion)
- Dequeue (Deletion)
- Peek (Front)
- isEmpty
- isFull
- Size

Implementations of Queues

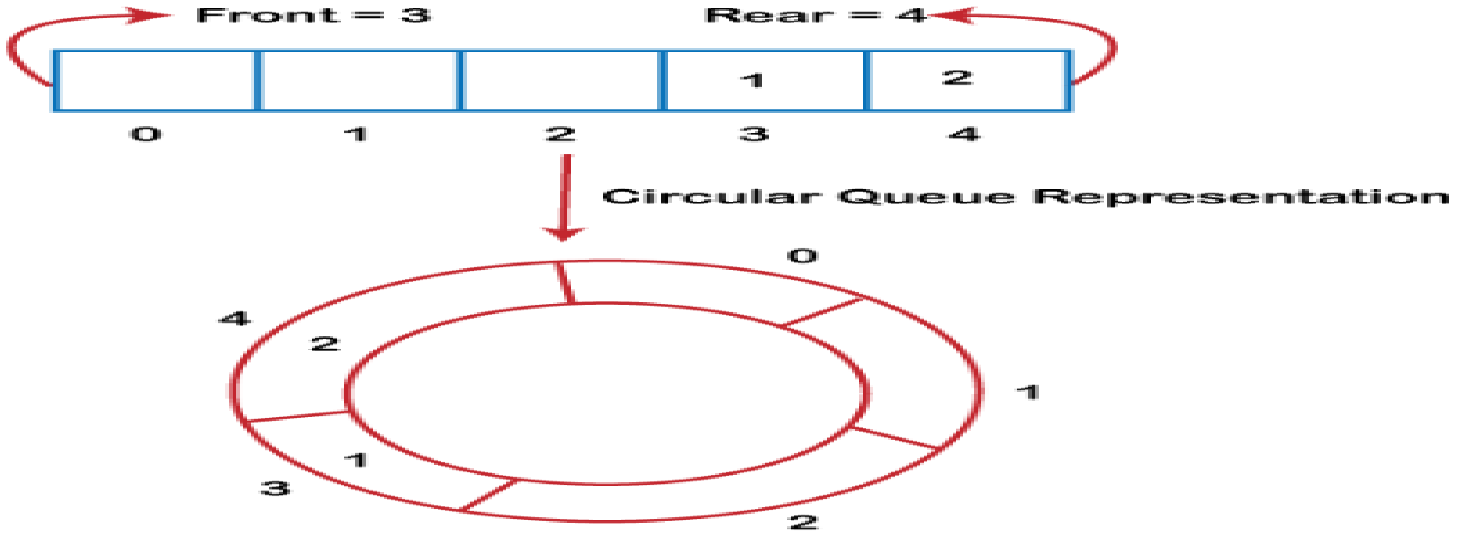
- Array-based Queue

Sessions 29 Outlook

- Circular Queues
- Operations
- Insertion and Deletion Algorithm
- DeQueue
- Assignment

Why was the circular queue?

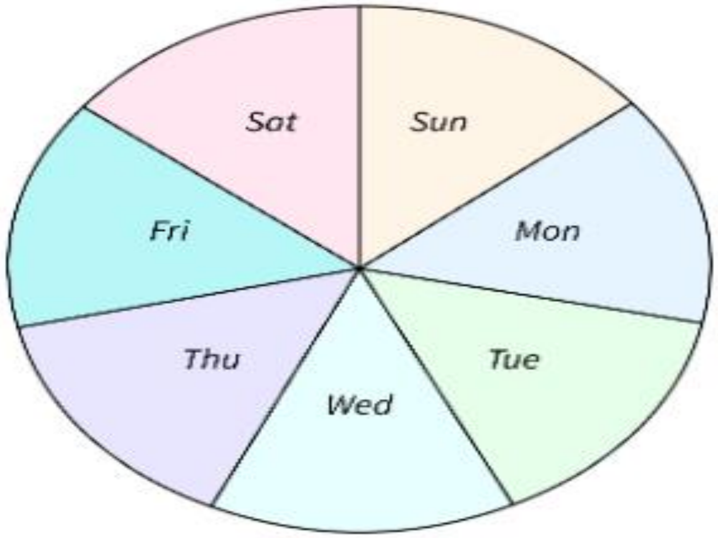
- If the rear reaches to the end position of the Queue then there might be possibility that some vacant spaces are left in the beginning which cannot be utilized. So, to overcome such limitations, the concept of the circular queue was introduced.



https://www.javatpoint.com/data-structure-queue

Circular Queue?

- It is also based on the FIFO (First In First Out) principle except that the last position is connected to the first position in a circular queue that forms a circle. It is also known as a ***Ring Buffer***..



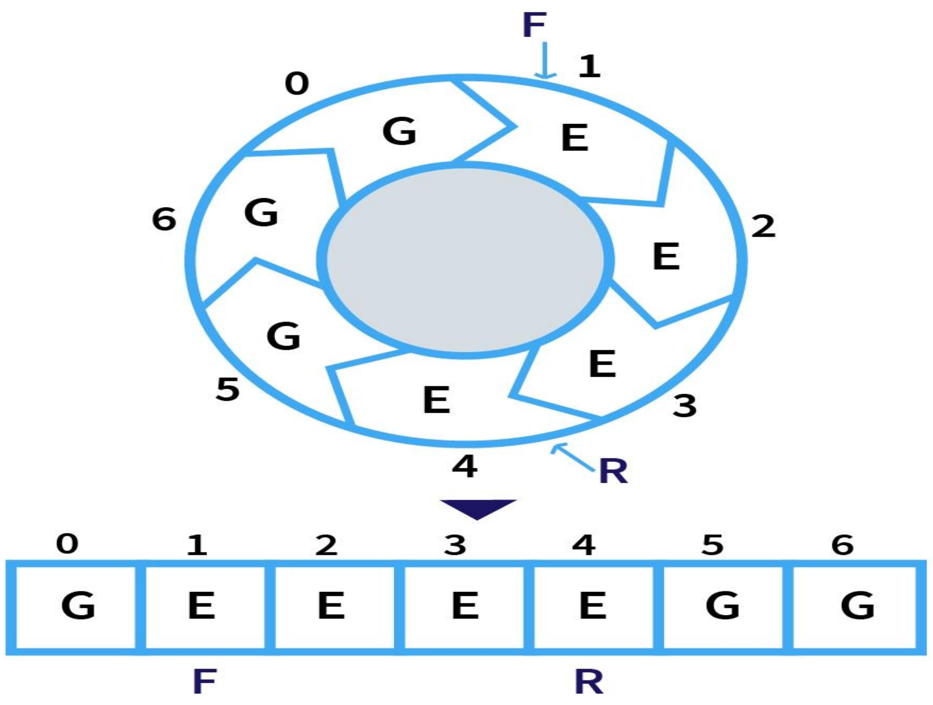
https://www.scaler.com/topics/data-structures/queue-in-data-structure/

Operations

- **Front:** It is used to get the front element from the Queue.
- **Rear:** It is used to get the rear element from the Queue.
- **EnQueue(value):** This function is used to insert the new value in the Queue. The new element is always inserted from the rear end.
- **DeQueue():** This function deletes an element from the Queue. The deletion in a Queue always takes place from the front end.

https://www.youtube.com/watch?app=desktop&v=gYxdHbbjn_8

Representation using Array



n (maxm size of queue) = 7
E = Elements / Data
G = Garbage / No Data

R > F, So
No of Elements present = R - F + 1 = 4 - 1 + 1 = 4

https://www.youtube.com/watch?app=desktop&v=gYxdHbbjn_8

Implement Circular Queue using Array:

Insertion

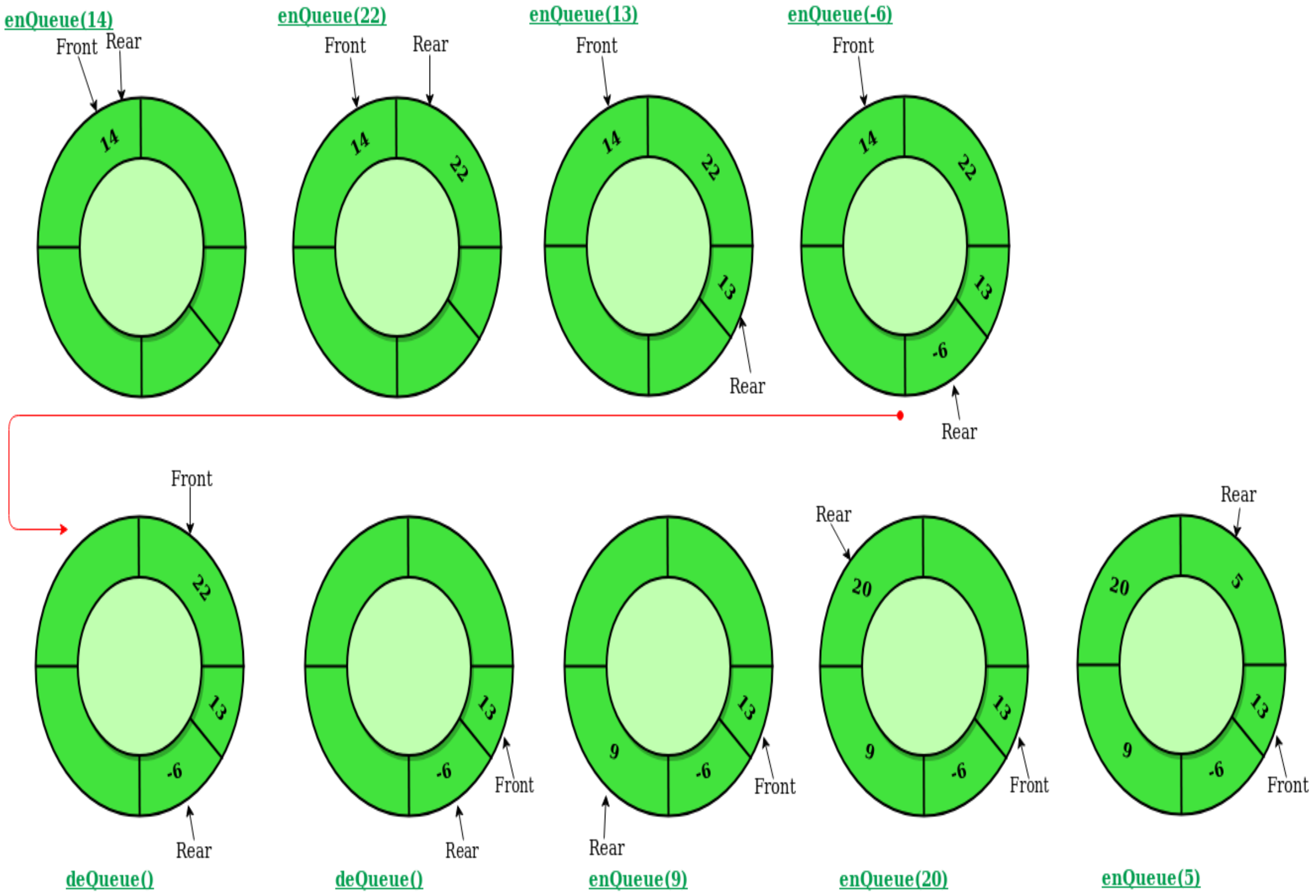
```
Step 1 : If FRONT = (REAR + 1) % MAXSIZE : then
 Write : “Queue Overflow” and return.
 [End of If structure]
Step 2 : Read NUM to be inserted in Circular Queue.
Step 3 : If FRONT= -1 : then
 Set FRONT = REAR =0.
 Else
 Set REAR=(REAR + 1) % MAXSIZE.
 [End of If Else structure]
Step 4 : Set CQUEUE[REAR]=NUM;
Step 5 : Exit
```

Implement Circular Queue using Array:

Deletion

```
Step 1 : If FRONT = - 1 : then
 Write : “Queue Underflow” and return.
 [End of If Structure]
Step 2 : Set NUM = CQUEUE[FRONT].
Step 3 : Write ‘Deleted element from circular queue is : ",NUM.
Step 4 : If FRONT = REAR : then
 Set FRONT = REAR = -1;
 Else
 Set FRONT = (FRONT + 1) % MAXSIZE.
Step 5 : Exit
```

Illustration of Circular Queue Operations:



https://www.geeksforgeeks.org/introduction-to-circular-queue/

Complexity Analysis of Circular Queue Operations

Time Complexity:

- Enqueue: O(1) because no loop is involved for a single enqueue.
- Dequeue: O(1) because no loop is involved for one dequeue operation.

Auxiliary Space: O(N) as the queue is of size N.

Advantages of Circular Queue Operations

Advantages:

Efficient space utilization: Circular queues efficiently use the available memory space.

No shifting of elements: Elements are continuously added and removed without the need to shift remaining elements.

Simple implementation: It's relatively straightforward to implement a circular queue using an array.

Disadvantage of Circular Queue Operations

Disadvantages:

Fixed size: Circular queues have a fixed capacity determined by the size of the underlying array.

Potential for confusion: The circular nature of the queue can sometimes lead to confusion in understanding its behavior, especially with wraparound scenarios.

Applications of Circular Queue Operations

Applications:

- Operating systems for managing task scheduling.
- Networking for managing packets in routers and switches.
- Memory management in embedded systems.
- Simulation of resource allocation scenarios.

Code for insertion &deletion Operation Of Queue

```
// C++ program for insertion and
// deletion in Circular Queue

#include <bits/stdc++.h>
using namespace std;

// Structure of a Node
struct Node {
    int data;
    struct Node* link;
};

struct Queue {
    struct Node *front, *rear;
};
```

```
// Function to create Circular queue
void enQueue(Queue* q, int value)
{
    struct Node* temp = new Node;
    temp->data = value;
    if (q->front == NULL)
        q->front = temp;
    else
        q->rear->link = temp;

    q->rear = temp;
    q->rear->link = q->front;
}
```

```
// Function to delete element from Circular Queue
int deQueue(Queue* q)
{
    if (q->front == NULL) {
        cout << "Queue is empty";
        return INT_MIN;
    }

    // If this is the last node to be deleted
    int value; // Value to be dequeued
    if (q->front == q->rear) {
        value = q->front->data;
        free(q->front);
        q->front = NULL;
        q->rear = NULL;
    }
else // There are more than one
nodes
    {
        struct Node* temp = q->front;
        value = temp->data;
        q->front = q->front->link;
        q->rear->link = q->front;
        free(temp);
    }

    return value;
}
```

```
// Function displaying the elements of
Circular Queue
void displayQueue(struct Queue* q)
{
    struct Node* temp = q->front;
    cout << endl << "Elements in Circular
Queue are: ";
    while (temp->link != q->front) {
        cout << temp->data << " ";
        temp = temp->link;
    }
    cout << temp->data;
}
```

```
/* Driver of the program */
int main()
{
    // Create a queue and initialize
front and rear
    Queue* q = new Queue;
    q->front = q->rear = NULL;

    // Inserting elements in Circular
Queue
    enQueue(q, 14);
    enQueue(q, 22);
    enQueue(q, 6);

    // Display elements present in
Circular Queue
    displayQueue(q);
```

```
// Deleting elements from Circular
Queue
    cout << endl << "Deleted value = " <<
deQueue(q);
    cout << endl << "Deleted value = " <<
deQueue(q);

    // Remaining elements in Circular
Queue
    displayQueue(q);

    enQueue(q, 9);
    enQueue(q, 20);
    displayQueue(q);

    return 0;
}
```

Output

```
Elements in Circular Queue are:

14 22 6

Deleted value = 14

Deleted value = 22

Elements in Circular Queue are: 6

 Elements in Circular Queue are: 6  9  20;
```

Test Yourself

The circular Queue of capacity (n - 1) is used in an array of n elements. Assume that insert and deletion functions are performed using REAR and FRONT as the variable index of Array, respectively. Initially, REAR = FORWARD = 0. The conditions for finding the Queue are full and empty.

a) (REAR+1) mod n == FRONT, empty: REAR == FRONT.
b) empty: (FRONT+1) mod n == REAR,(REAR+1) mod n == FRONT
c) empty: (REAR+1) mod n == FRONT, REAR == FRONT
d) (FRONT+1) mod n == REAR, empty: REAR == FRONT.

Solution: a. (REAR+1) mod n == FRONT, empty: REAR == FRONT

A standard circular Queue 'q' implemented whose size is 11 from 0 to 10. The front & rear pointers start to point at q[2]. The ninth element be added at which position?

a) q[0]
b) q[1]
c) q[9]
d) q[10]

Solution: a. q[0]

Choose the incorrect statement?

a) If Queue is implemented through the linked list, keeping track of a front pointer will change only rear pointer "s" during insertion into non-empty Queue.

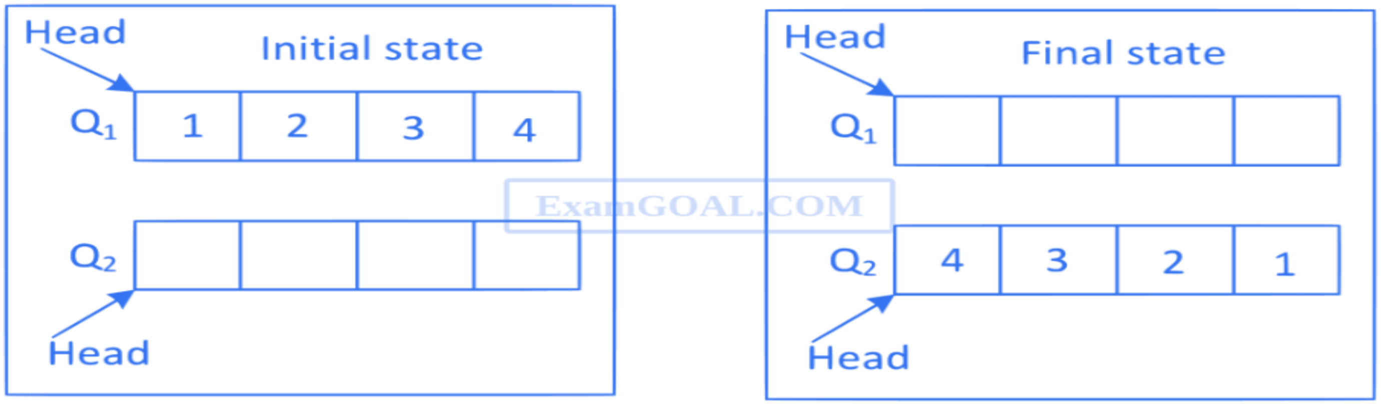
b) The Queue can be used to implement the least recently used page fault algorithm and Quicksort algorithm.

c) The Queue can be used to implement the Quicksort algorithm but not least recently used (LRU) page fault algorithm.

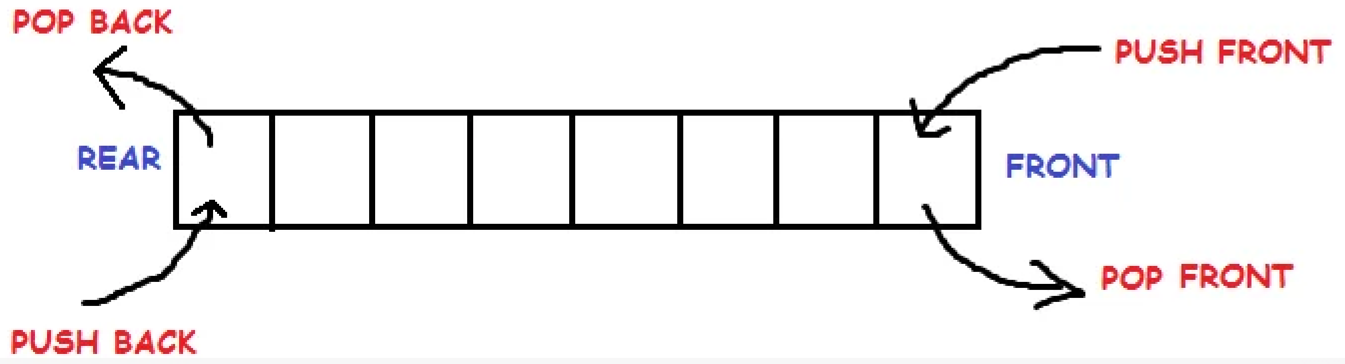
d) Both (A) and (C)

Solution: c. The Queue can be used to implement the Quicksort algorithm but not least recently used (LRU) page fault algorithm.

Consider the queues Q_1 containing four elements and Q_2 containing none (shown as the Initial State in the figure). The only operations allowed on these two queues are Enqueue (Q, element) and Dequeue (Q). The minimum number of Enqueue operations on Q_1 required to place the elements of Q_1 in Q_2 in reverse order (shown as the Final State in the figure) without using any additional storage is _______________.



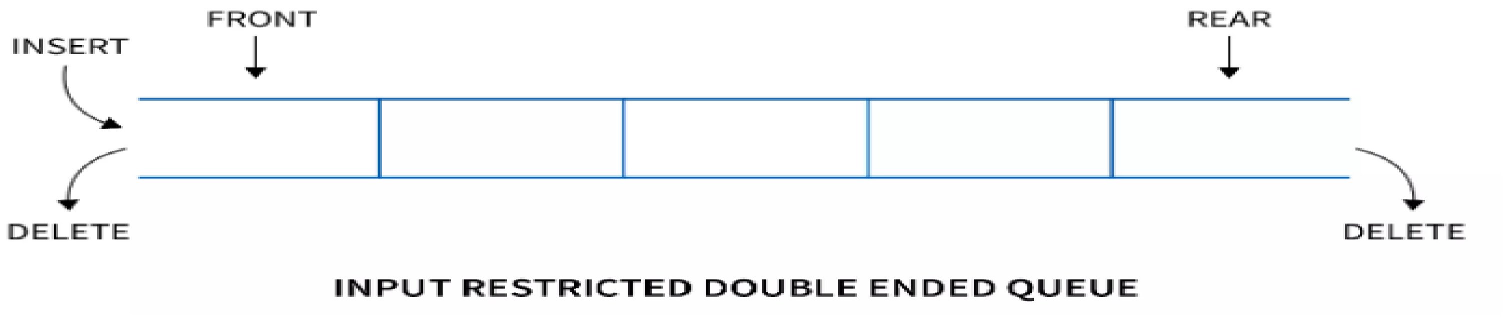
Doubly Ended Queue



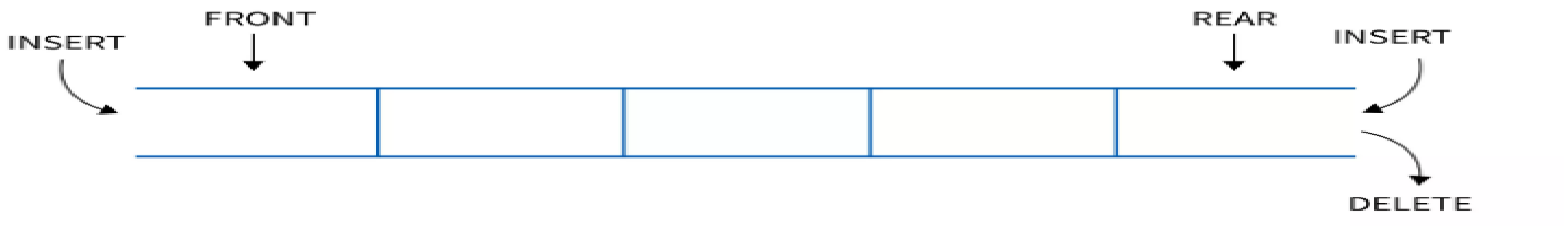
Doubly Ended Queue

Types

- Input restricted queue
- Output restricted queue



INPUT RESTRICTED DOUBLE ENDED QUEUE



OUTPUT RESTRICTED DOUBLE ENDED QUEUE

Operations performed on deque

There are the following operations that can be applied on a deque -

- Insertion at front
- Insertion at rear
- Deletion at front
- Deletion at rear
- Get the front item from the deque
- Get the rear item from the deque
- Check whether the deque is full or not
- Checks whether the deque is empty or not

Assignmentunit2 _1

1. Convert the given expression from infix to postfix
   1. (a + b) * (c + d * e) * f
   2. a – b * c / d + e

2. Evaluate the given postfix expression by using stack concept
   1. i) 5 5 2 3 + 6 * + 3 + *
   2. ii) 10 8 + 5 * - 2

3.
   a) Write the pseudo code for Push and POP operation from Stack. Give with suitable example.

   b) Write the pseudo code for inserting and deletion from Queue.

Assignmentunit2 _1

4. Fill in the blank

a) In a stack, the operation of removing an item is called ________.

b) The basic principle of ________ is Last In, First Out (LIFO).

5. Consider a stack that initially contains 5 elements. If you perform 3 push operations and 2 pop operations, what is the current size of the stack?

6. If you have a queue with 10 elements and you perform 5 dequeue operations followed by 7 enqueue operations, how many elements does the queue contain now?

Assignmentunit2 _1

7. One way to implement a queue is to use a circular linked list. In a circular linked list, the last node's next pointer points at the first node. Assume the list does not contain a header and that we can maintain, at most, one iterator corresponding to a node in the list. For which of the following representations can all basic queue operations be performed in constant worst-case time? Justify your answers.
   a. Maintain an iterator that corresponds to the first item in the list.
   b. Maintain an iterator that corresponds to the last item in the list

Assignmentunit2 _1

8. Draw the queue structure in each case when the following operations are performed on an empty queue.

(a)	Add A, B, C, D, E, F
(b)	Delete two letters
(c)	Add G
(d)	Add H
(e)	Delete four letters
(f)	Add I

Assignmentunit2 _1

9. Consider the following sequence of operations on an empty stack.

push(54); push(52); pop(); push(55); push(62); **s** = pop();

Consider the following sequence of operations on an empty queue.

enqueue(21); enqueue(24); dequeue(); enqueue(28); enqueue(32); **q** = dequeue();

The value of s + q is ____

Assignmentunit2 _1

10. Consider the following Circular queue where CIRCULAR QUEUE is allocated 6 memory cells: Front=2, Rear=5

QUEUE: ___________, London, Berlin, Rome, Paris, __________

Describe the Queue including FRONT and REAR as the following operations take place:

- Athens is added
- Two cities are deleted
- Madrid is added
- Moscow is added
- Three cities are deleted
- Oslo is added

Write an algorithm to perform insertion of new element in Circular Queue?

Review

Introduction: Circular queues represent a key data structure leveraging a fixed-size array with a circular design. This design facilitates efficient management of data with a continuous wraparound structure.

Key Concepts:

Circular Behavior: Rear and front pointers wrap around the array, enabling a circular structure that optimizes space utilization.

FIFO Principle: Circular queues adhere to the First In, First Out principle, ensuring that the earliest enqueued elements are dequeued first.

Review

Operations

Enqueue (Insertion): Addition of elements at the rear of the queue.

Dequeue (Deletion): Removal of elements from the front of the queue.

isFull and isEmpty: Checks for fullness and emptiness of the queue, respectively.

Front and Rear: Access to the front and rear elements without modification.

THANK
YOU