Queues from Linked Lists

Week 9

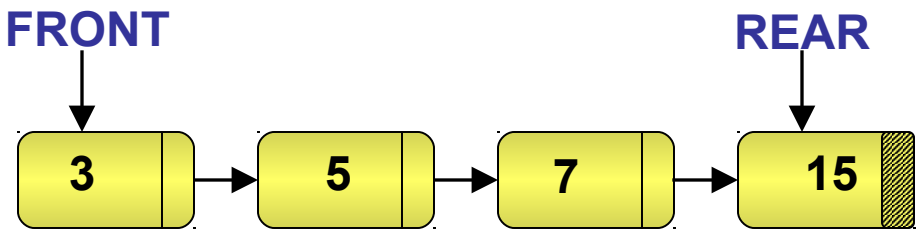
Lecture 3

Objectives

- In this session, you will learn to:
  - Linked List implementation of Queues

Implementing a Queue Using a Linked List

- To keep track of the rear and front positions, you need to declare two variables/pointers, REAR and FRONT, that will always point to the rear and front end of the queue respectively.
- If the queue is empty, REAR and FRONT point to NULL.



Inserting an Element in a Linked Queue

- Write an algorithm to implement insert operation in a linked queue.

Inserting an Element in a Linked Queue (Contd.)

- Algorithm to insert an element in a linked queue.

- Suppose initially, the queue is empty.

Request number generated **3**

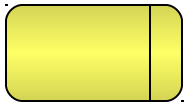
REAR = NULL
FRONT = NULL

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

REAR = NULL
FRONT = NULL



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
    a. Make FRONT point to the new node
    b. Make REAR point to the new node
    c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

REAR = NULL
FRONT = NULL

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
    a. Make FRONT point to the new node
    b. Make REAR point to the new node
    c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

REAR = NULL
FRONT = NULL

1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
    a. Make FRONT point to the new node
    b. Make REAR point to the new node
    c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

REAR = NULL
FRONT = NULL

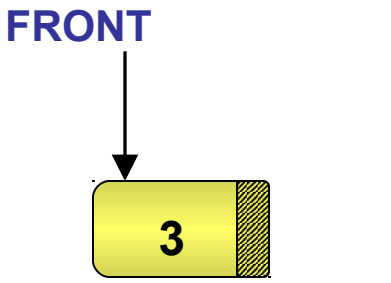
1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
    a. Make FRONT point to the new node
    b. Make REAR point to the new node
    c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

REAR = NULL

FRONT = NULL

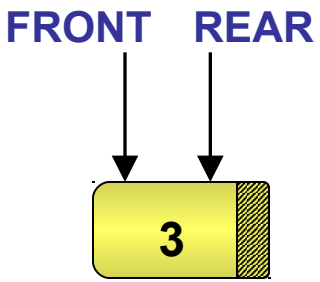


1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

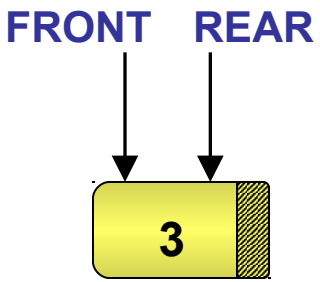
REAR = NULL



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **3**

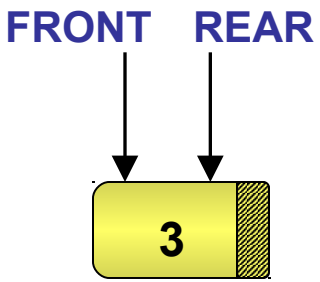


1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Insert operation complete

Inserting an Element in a Linked Queue (Contd.)

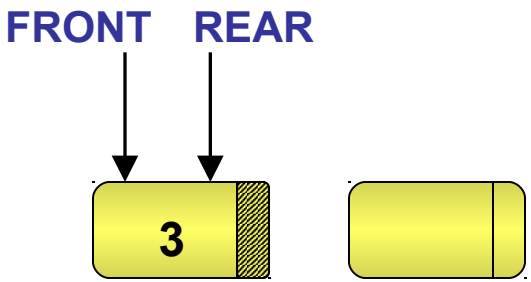
Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

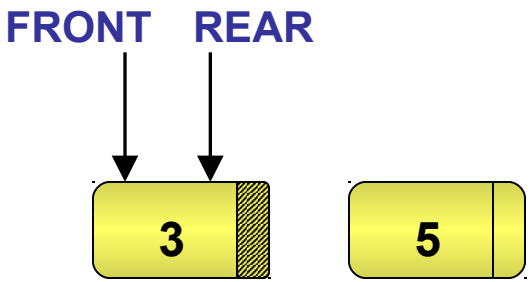
Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

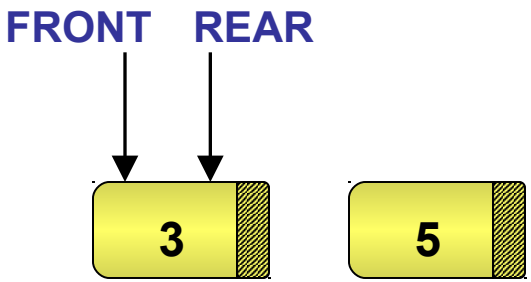
Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

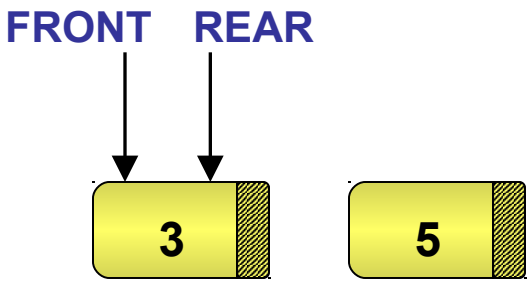
Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

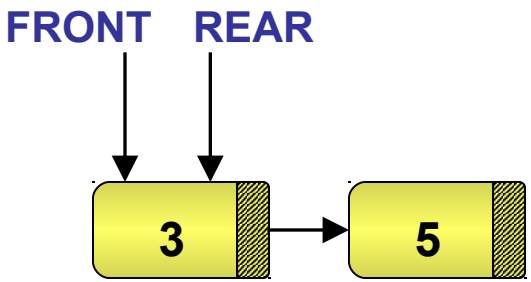
Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

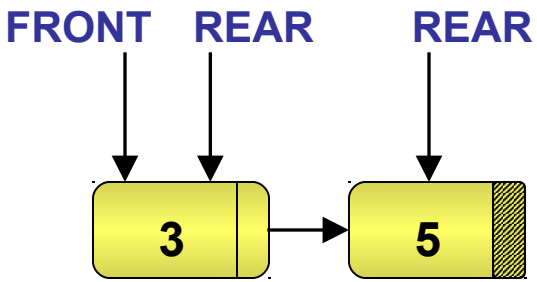
Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Inserting an Element in a Linked Queue (Contd.)

Request number generated **5**



1. Allocate memory for the new node.
2. Assign value to the data field of the new node.
3. Make the next field of the new node point to NULL.
4. If the queue is empty, execute the following steps:
   a. Make FRONT point to the new node
   b. Make REAR point to the new node
   c. Exit
5. Make the next field of REAR point to the new node.
6. Make REAR point to the new node.

Insert operation complete

ALGORITHM TO IMPLEMENT INSERT OPERATION

FRONT= NULL

REAR = NULL

Algorithm Insert()

{

1. Create node [(new1 = (struct node*) malloc(sizeof(struct node))]

2. Enter data [new1 -> info =data]

3.if(FRONT == NULL)

        3.1 FRONT = new1

        3.2 REAR = new1

   else

        3.1 REAR -> next = new1

        3.2 REAR = new1

4. REAR ->next = NULL

}

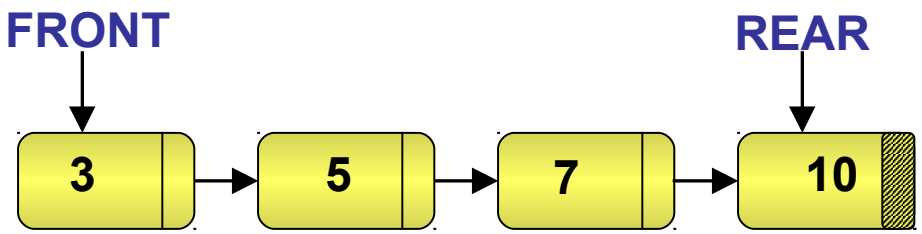
Deleting an Element from a Linked Queue

- Write an algorithm to implement the delete operation on a linked queue.

Deleting an Element from a Linked Queue (Contd.)

- Algorithm to implement delete operation on a linked queue.

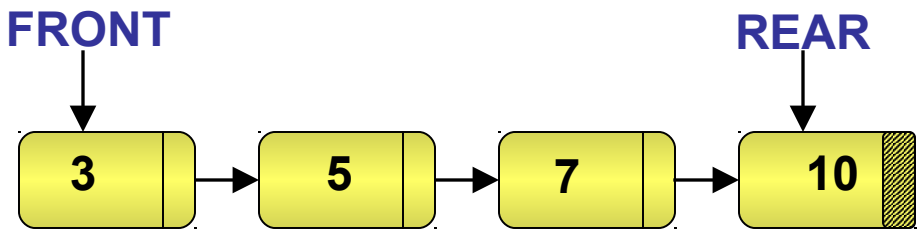
One request processed



1. If the queue is empty: // **FRONT = NULL**
   a. Display “Queue empty”
   b. Exit
2. Mark the node marked FRONT as current
3. Make FRONT point to the next node in its sequence
4. Release the memory for the node marked as current

Deleting an Element from a Linked Queue (Contd.)

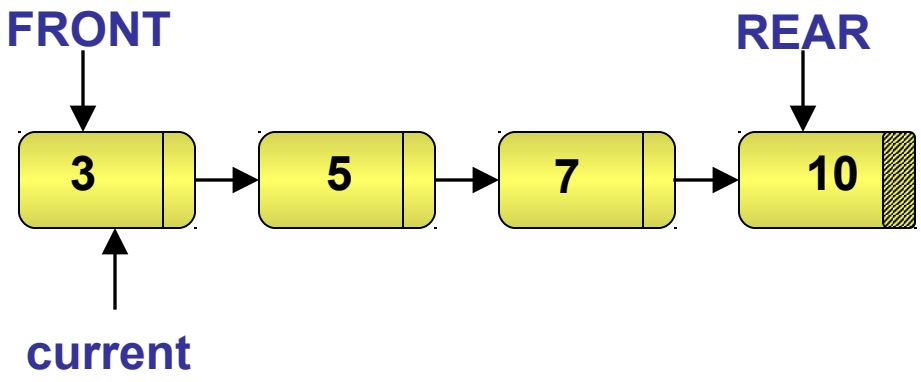
One request processed



1. If the queue is empty: // **FRONT = NULL**
   a. Display “Queue empty”
   b. Exit
2. Mark the node marked FRONT as current
3. Make FRONT point to the next node in its sequence
4. Release the memory for the node marked as current

Deleting an Element from a Linked Queue (Contd.)

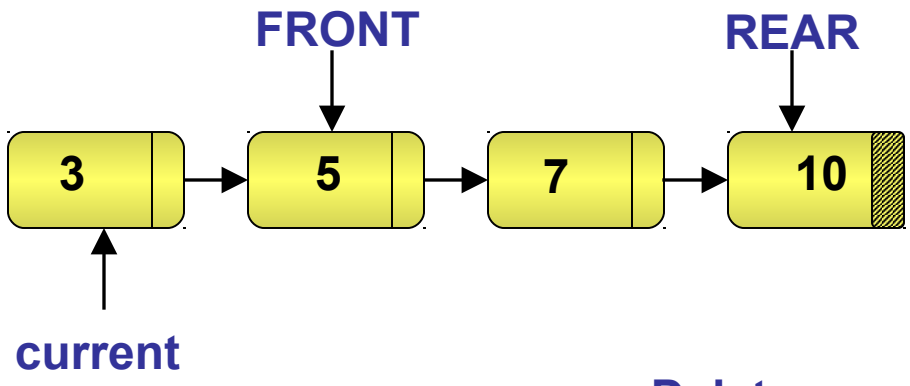
One request processed



1. If the queue is empty: // **FRONT = NULL**
   a. Display “Queue empty”
   b. Exit
2. Mark the node marked FRONT as current
3. Make FRONT point to the next node in its sequence
4. Release the memory for the node marked as current

Deleting an Element from a Linked Queue (Contd.)

One request processed



1. If the queue is empty: **// FRONT = NULL**
   a. Display “Queue empty”
   b. Exit
2. Mark the node marked FRONT as current
3. Make FRONT point to the next node in its sequence
4. Release the memory for the node marked as current

Delete operation complete

Memory released

ALGORITHM TO IMPLEMENT THE DELETE OPERATION

```
Algorithm Delete()
{
1. If (FRONT == 0)
        1.1 Print "underflow“
   else
        1.1 Current = FRONT
        1.2 FRONT = FRONT -> next
        1.3 Current -> next = NULL
        1.4 Release the memory [ free (Current) ]

}
```