Queues

Kuan-Yu Chen (陳冠宇)

2018/10/03 @ TR-212, NTUST

Review

- Stack
  - FILO

- Algorithms to covert
  - Infix to Postfix
  - Infix to Prefix

- Queue
  - FIFO

Homeork1: Expression Convertor.

- Given a infix expression, please convert the expression to **both** prefix and postfix expressions
  - The implementation **must** base on stack
  - Please show the results **step-by-step**
  - Please upload your source codes and a paper report to moodle
  - TA will ask you to demo your program
  - The **hard deadline** is 2018/10/15 8:00

Infix Scanned	Stack	Postfix Expression
	(	
A	(	A
-	(-	A
(	(- (	A
B	(- (	A B
/	(- (/	A B
C	(- (/	A B C

Homeork1: Expression Convertor..

- Given a infix expression, please convert the expression to **both** prefix and postfix expressions
  - The length of the input expression will always less than 30
  - Only five operators need to be considered
    - +, −, ×, ÷, %
  - The operands are capital letters (i.e., A~Z)

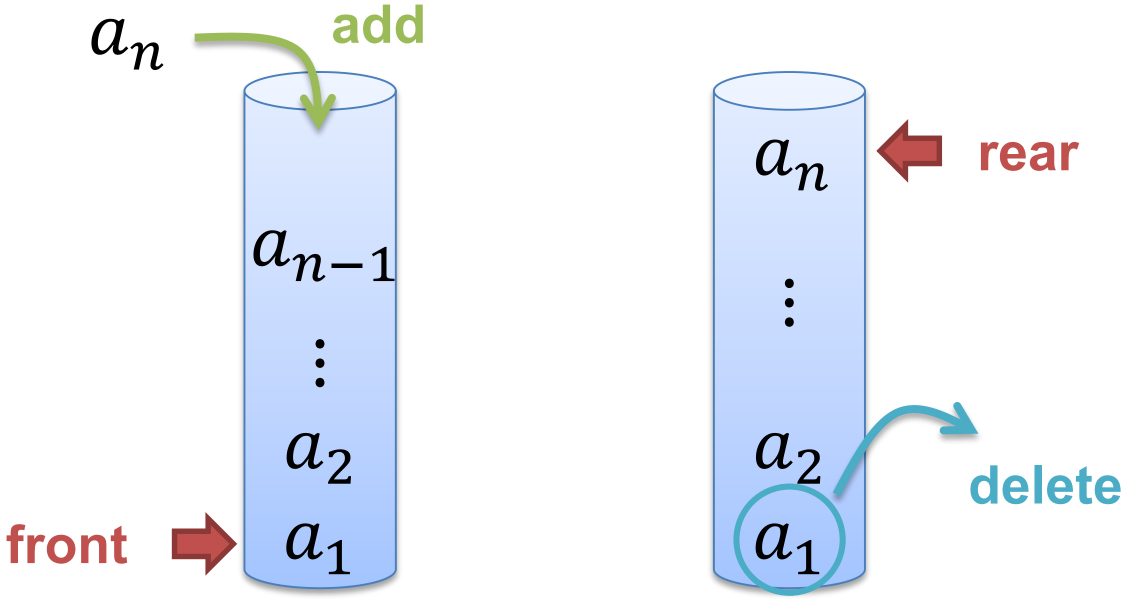
Infix Scanned	Stack	Postfix Expression
	(	
A	(	A
-	(-	A
(	(- (	A
B	(- (	A B
/	(- (/	A B
C	(- (/	A B C

Homeork1: Expression Convertor..

- Given a infix expression, please convert the expression to **both** prefix and postfix expressions
  - $(A - B \div C) \times (A \div K - L)$
    - Prefix: $\times -A \div BC -\div AKL$
    - Postfix: $ABC \div -AK \div L -\times$
  - $A - (B \div C + (D\%E \times F) \div G) \times H$
    - Prefix: $-A \times + \div BC \div \times \%DEFGH$
    - Postfix: $ABC \div DE\%F \times G \div +H \times -$

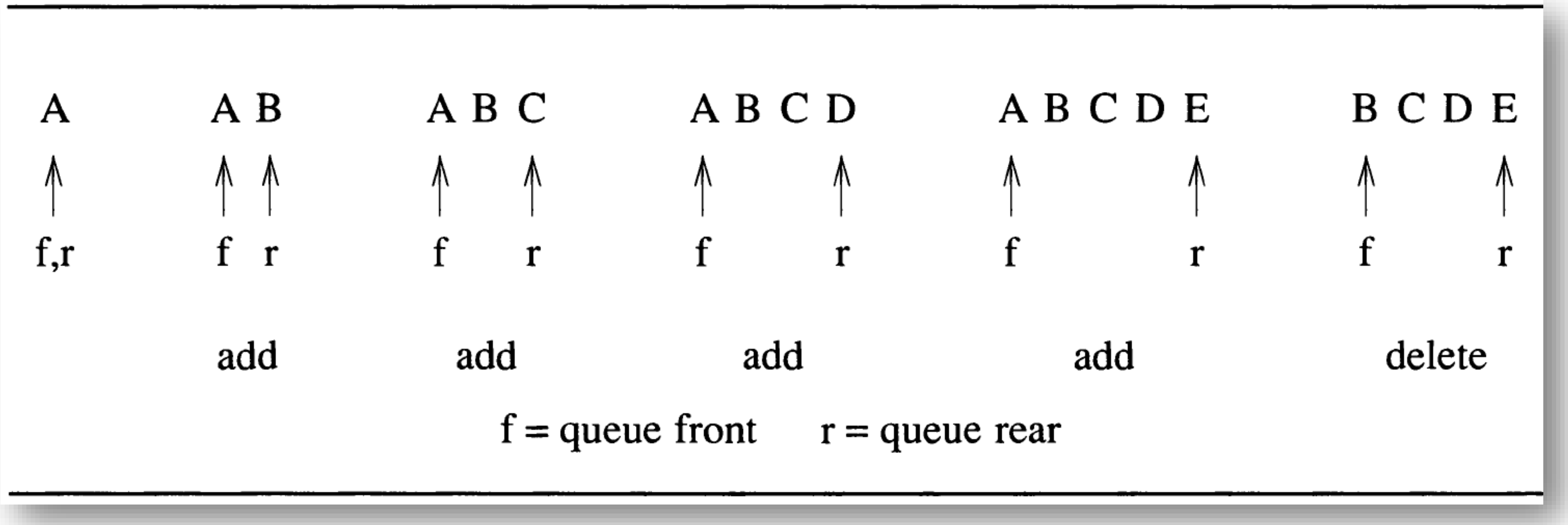
Queue.

- A **queue** is an **ordered** list in which insertions take place at one end (**rare**) and deletions are made at the opposite end (**front**)
  - Given a queue $Q = (a_1, a_2, \ldots, a_n)$
    - a_1 is the front element
    - a_n is the rear element
    - a_i is behind element a_{i-1}



Queue..

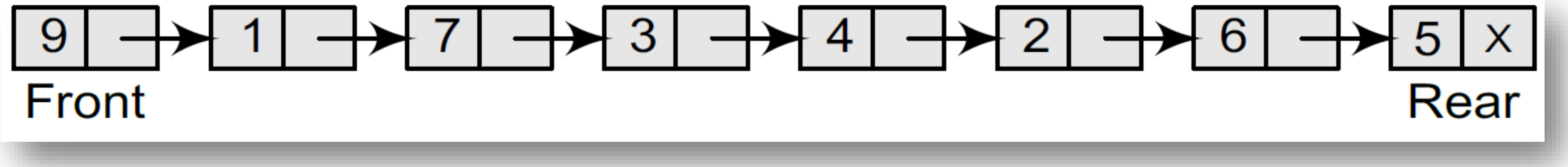
- By the definition of queue, if we insert the elements A, B, C, D, E in the order, then A is the first element deleted from the queue
  - **First-In-First-Out**

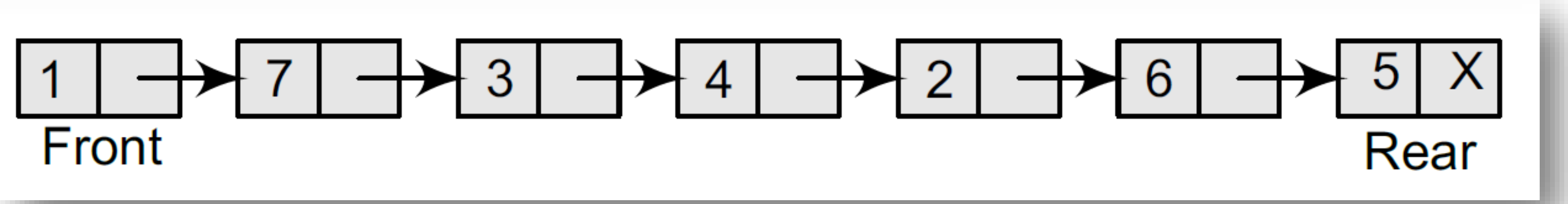


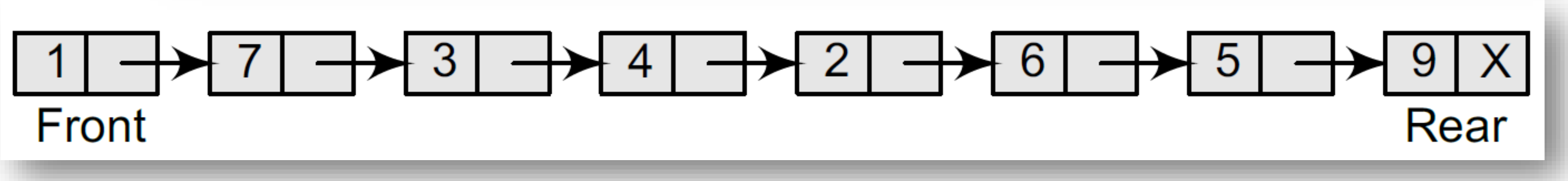
f = queue front r = queue rear

Implementation for Queue by Link List.

- Although creating a queue by an array is easy, its drawback is that the array must be declared to have some fixed size
- If the array size cannot be determined in advance, the linked representation is used

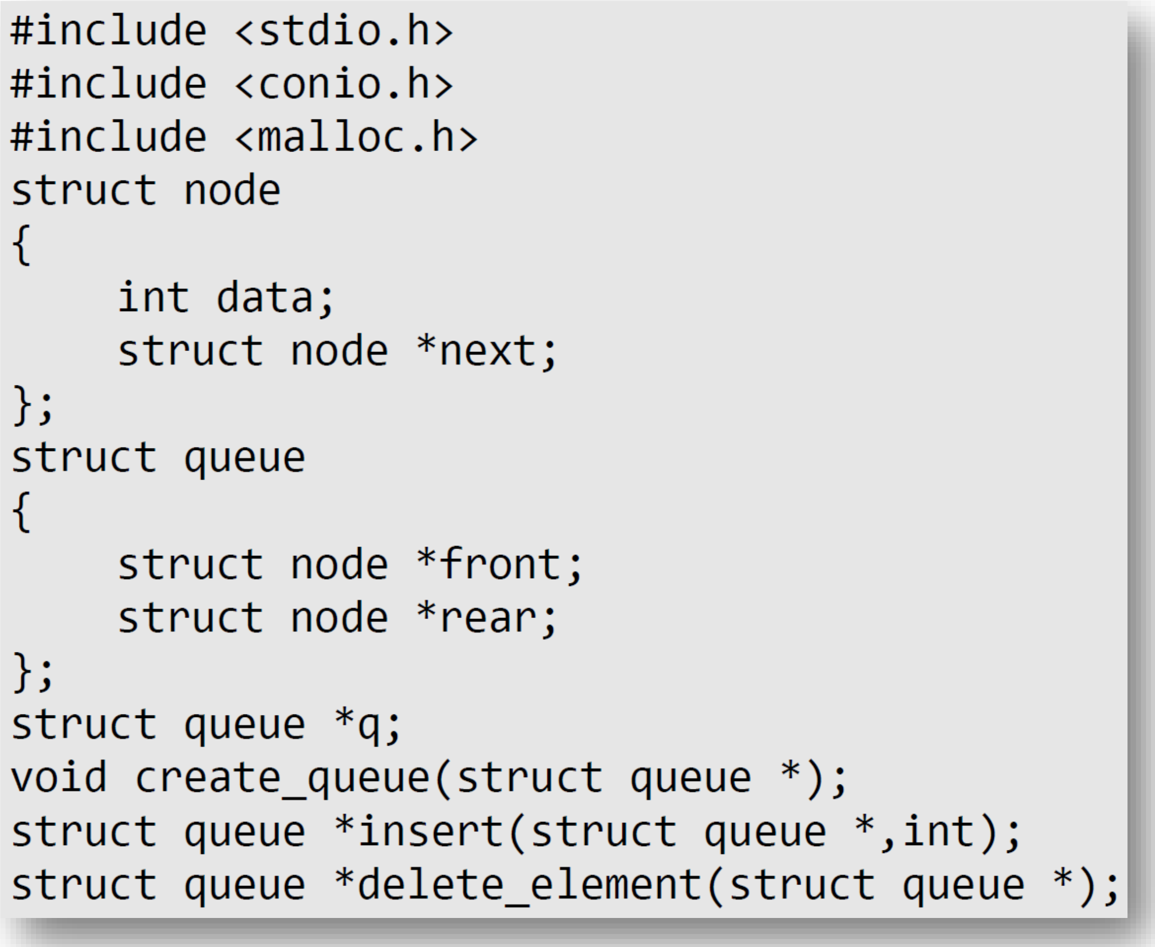






Implementation for Queue by Link List..

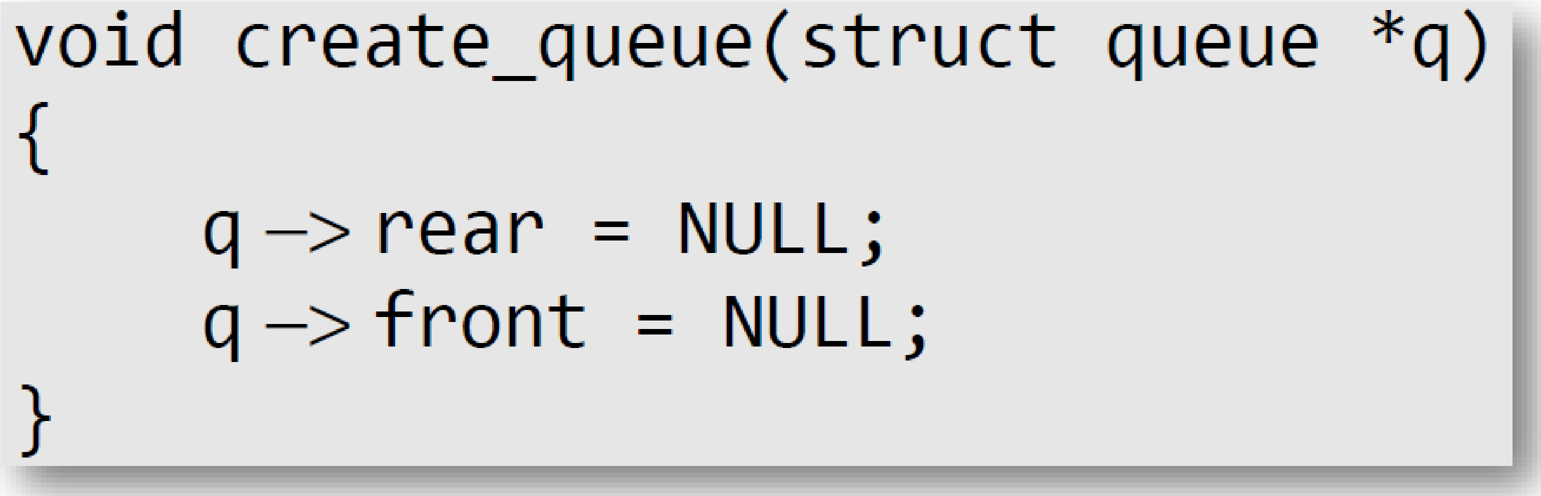
- Declare



```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node
{
    int data;
    struct node *next;
};
struct queue
{
    struct node *front;
    struct node *rear;
};
struct queue *q;
void create_queue(struct queue *);
struct queue *insert(struct queue *,int);
struct queue *delete_element(struct queue *);
```

Implementation for Queue by Link List...

- Create a queue

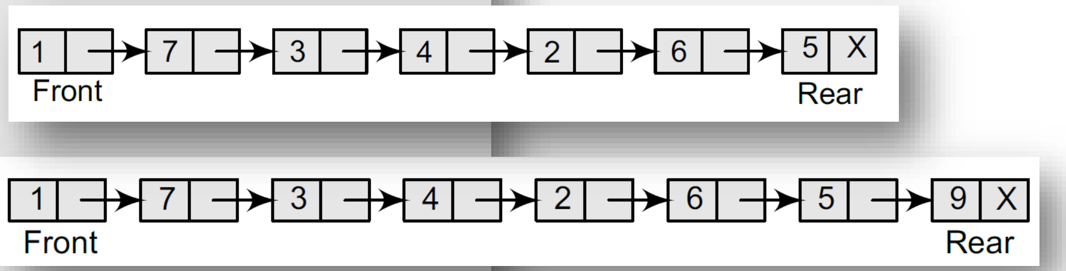


```
void create_queue(struct queue *q)
{
    q -> rear = NULL;
    q -> front = NULL;
}
```

Implementation for Queue by Link List....

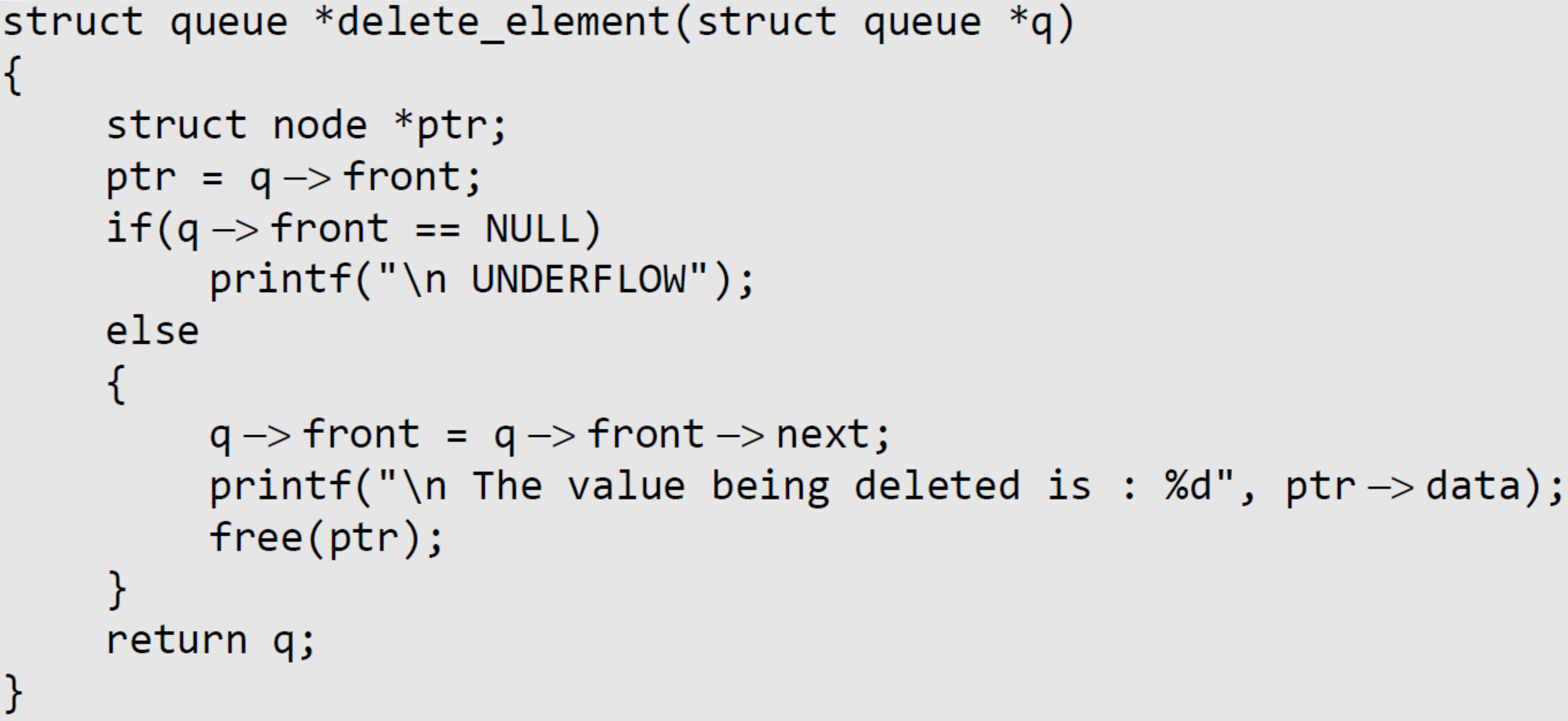
- For insertion

```
struct queue *insert(struct queue *q,int val)
{
    struct node *ptr;
    ptr = (struct node*)malloc(sizeof(struct node));
    ptr -> data = val;
    if(q -> front == NULL)
    {
        q -> front = ptr;
        q -> rear = ptr;
        q -> front -> next = q -> rear -> next = NULL;
    }
    else
    {
        q -> rear -> next = ptr;
        q -> rear = ptr;
        q -> rear -> next = NULL;
    }
    return q;
}
```

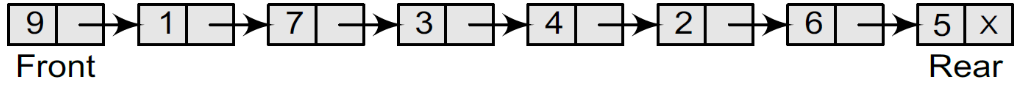


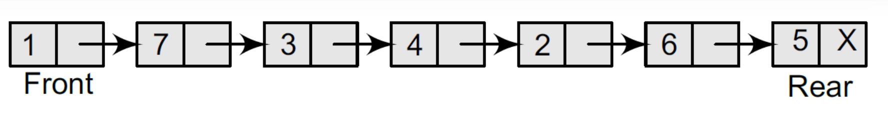
Implementation for Queue by Link List.....

- For deletion



```
struct queue *delete_element(struct queue *q)
{
    struct node *ptr;
    ptr = q -> front;
    if(q -> front == NULL)
        printf("\n UNDERFLOW");
    else
    {
        q -> front = q -> front -> next;
        printf("\n The value being deleted is : %d", ptr -> data);
        free(ptr);
    }
    return q;
}
```



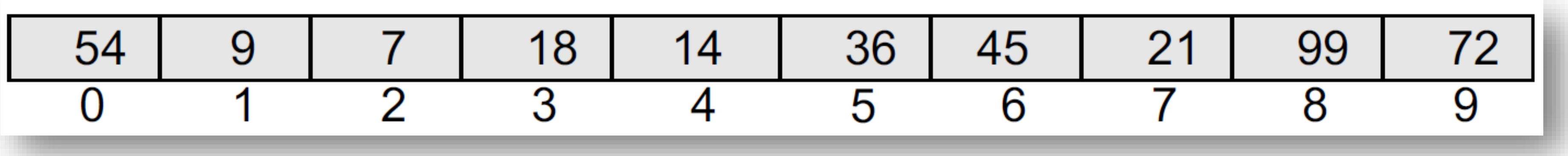


Types of Queues

- Actually a queue structure can be classified into four types
  - Circular Queue
  - Deque
  - Priority Queue
  - Multiple Queue

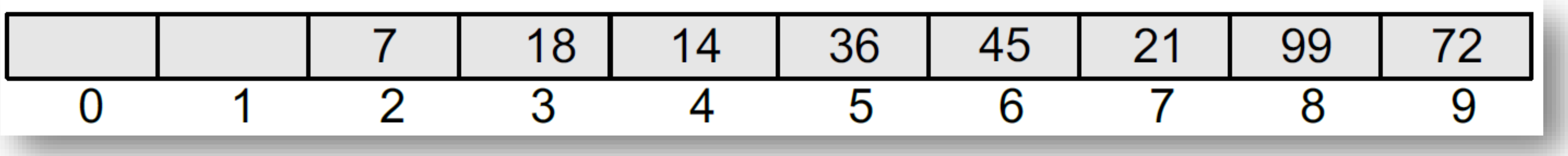
Circular Queue.

- Given a queue
  - if you want to insert another value, it will not be possible because the queue is completely full



54	9	7	18	14	36	45	21	99	72
0	1	2	3	4	5	6	7	8	9

  - let's delete two elements from the queue

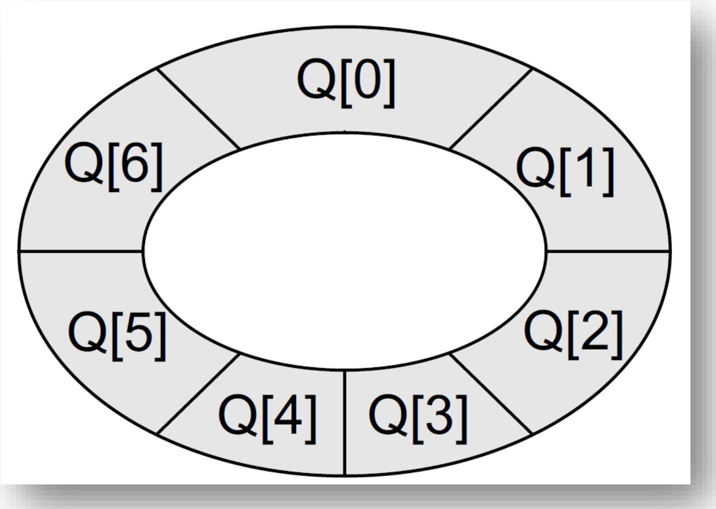


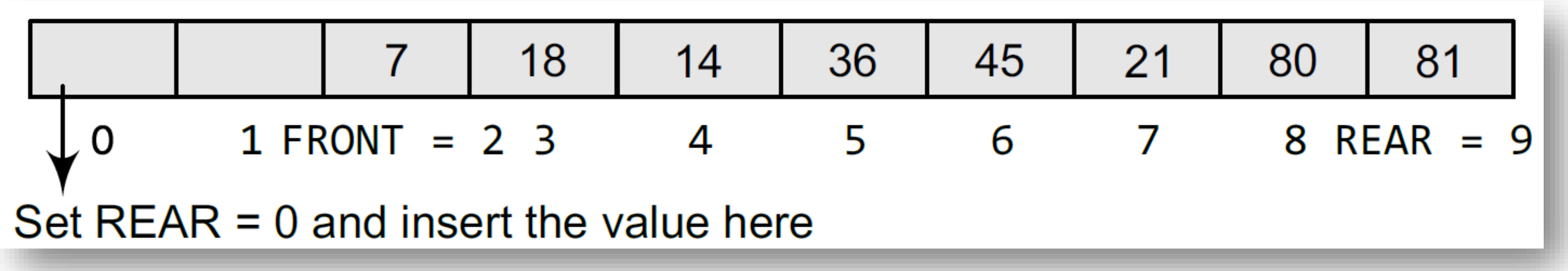
		7	18	14	36	45	21	99	72
0	1	2	3	4	5	6	7	8	9

  - Even though there is space available, we still can not insert elements in the queue
    1. Shift the elements to the left so that the vacant space can be occupied and utilized efficiently
    2. Circular queue!

Circular Queue..

- A circular queue is implemented by using array





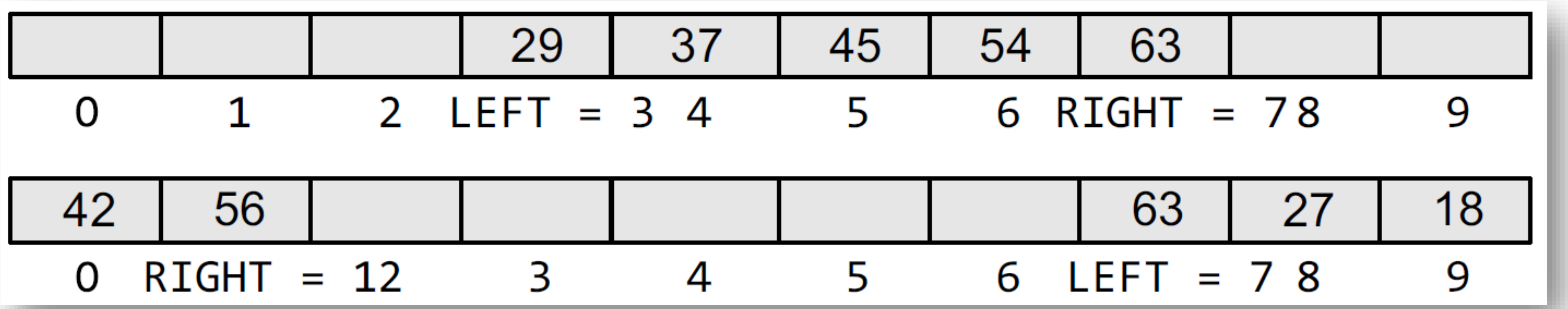
		7	18	14	36	45	21	80	81
0	1	FRONT = 2	3	4	5	6	7	8	REAR = 9

Set REAR = 0 and insert the value here

Deque.

- A deque (pronounced as “deck” or “dequeuer”) is a list in which the elements can be inserted or deleted at either end
  - Double-ended queue
  - It is also known as a **head-tail linked list** because elements can be added to or removed from either the front (head) or the back (tail) end
  - No element can be added and deleted from the middle

- In a deque, two pointers are maintained, **LEFT** and **RIGHT**, which point to either end of the deque



			29	37	45	54	63		
0	1	2	LEFT = 3	4	5	6	RIGHT = 7	8	9

42	56						63	27	18
0	RIGHT = 1	2	3	4	5	6	LEFT = 7	8	9

Deque..

- There are two variants of a double-ended queue
  - Input restricted deque
    - In this queue, insertions can be done only at one of the ends, while deletions can be done from both ends
  - Output restricted deque
    - In this queue, deletions can be done only at one of the ends, while insertions can be done on both ends

Priority Queue.

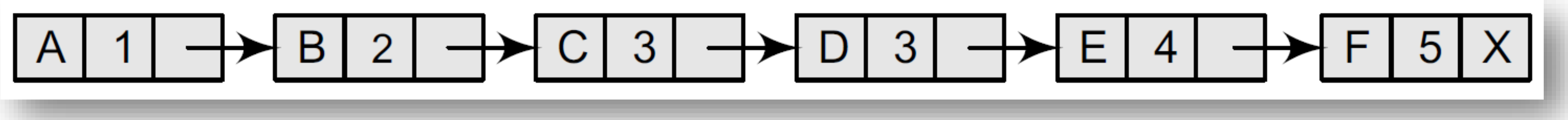
- A priority queue is a data structure in which each element is assigned a priority
  - The priority of the element can be set based on various factors

- The priority of the element will be used to determine the order in which the elements will be processed
  - An element with higher priority is processed before an element with a lower priority
  - Two elements with the same priority are processed on a first-come-first-served (FCFS) basis

- Priority queues are widely used in operating system
  - The priority of the process may be set based on the CPU time it requires to get executed completely
  - Brake override system, BOS

Priority Queue..

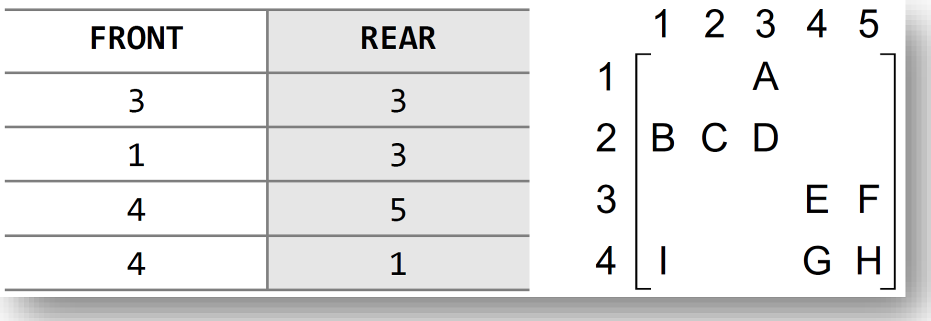
- Linked Representation of a Priority Queue
  - Every node of the list will have three parts:
    1. the information or data part
    2. the priority number of the element
    3. the address of the next element



  - From the example
    - Since A has a priority number 1 and B has a priority number 5, then A will be processed before B as it has higher priority than B
    - We cannot make out whether A was inserted before E or whether E joined the queue before A
    - We can definitely say that C was inserted in the queue before D because when two elements have the same priority

Priority Queue…

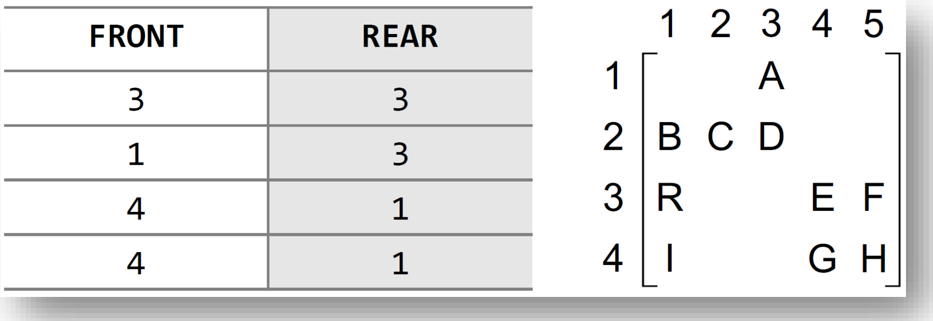
- Array Representation of a Priority Queue
  - Each priority number has its own queue
    - The queue is usually implemented by circular queue
  - Every individual queue will have its own FRONT and REAR pointers



FRONT	REAR
3	3
1	3
4	5
4	1

$$\begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & & & A & & \\ 2 & B & C & D & & \\ 3 & & & & E & F \\ 4 & I & & & G & H \end{array}$$

- If we have to insert an element R with priority number 3, then the priority queue will become



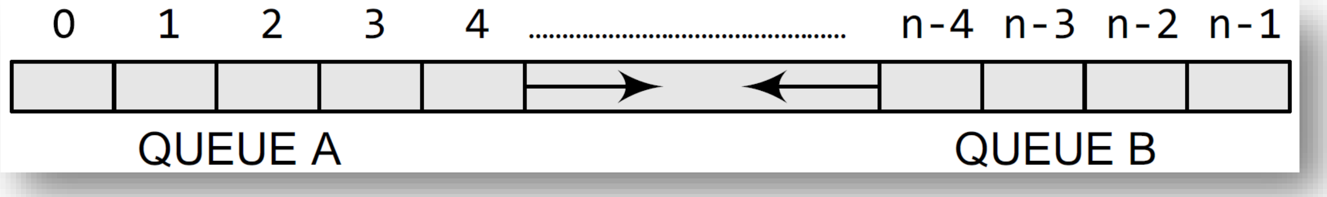
FRONT	REAR
3	3
1	3
4	1
4	1

$$\begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ 1 & & & A & & \\ 2 & B & C & D & & \\ 3 & R & & & E & F \\ 4 & I & & & G & H \end{array}$$

Multiple Queue.

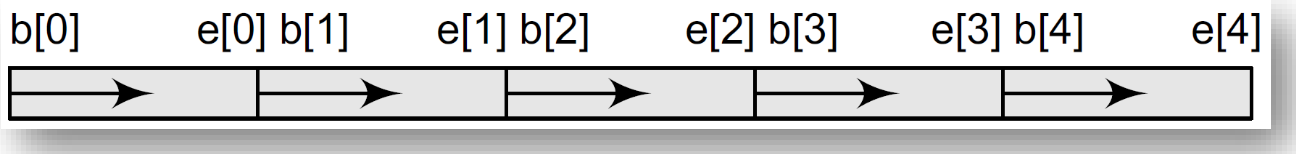
- When we implement a queue using an array, the size of the array must be known in advance
  - In case we allocate a large amount of space for the queue, it will result in sheer wastage of the memory

- A better solution to deal with this problem is to have multiple queues or to have more than one queue in the same array of sufficient size
  - Queue *A* will grow from left to right, whereas queue *B* will grow from right to left at the same time

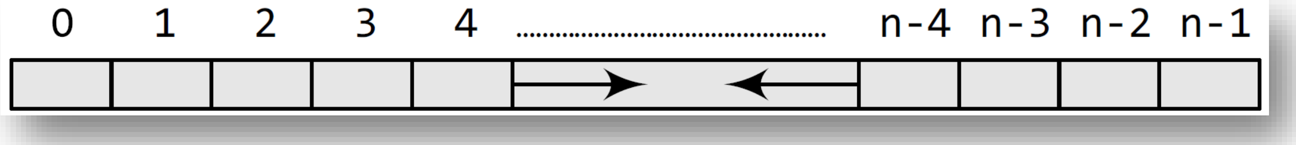


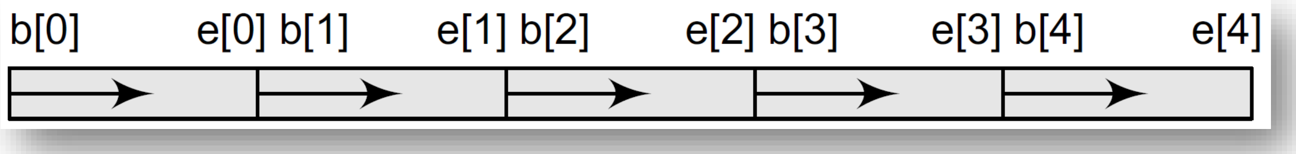
Multiple Queue..

- Extending the concept to multiple queues, another multiple queue has been introduce



- The concept of multiple queue can be extended to implement the “multiple stack”





Questions?

kychen@mail.ntust.edu.tw