**Stack**

* Last in, First out
* All operations are made only at the end of the data.
* Therefore, the time complexity for all operations is O(1)
* The location where the operation takes place is called top, and the insertion is push, and the deletion is pop.

**Stack Application**

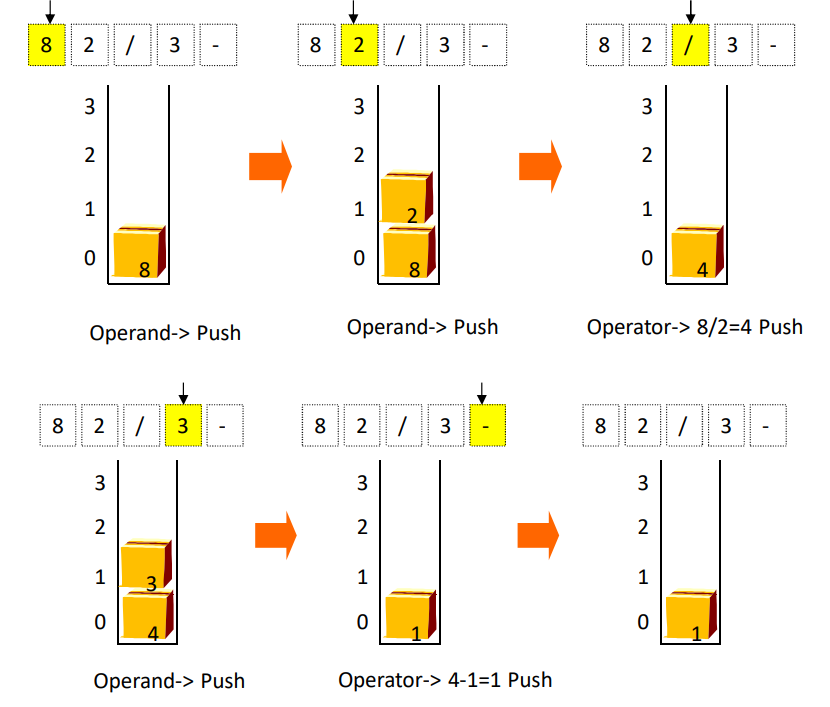
1. parenthesis Check

Types of parentheses – brackets: ‘[’, ‘]’ – braces: ‘{’, ‘}’ – parentheses: ‘(’, ‘)’

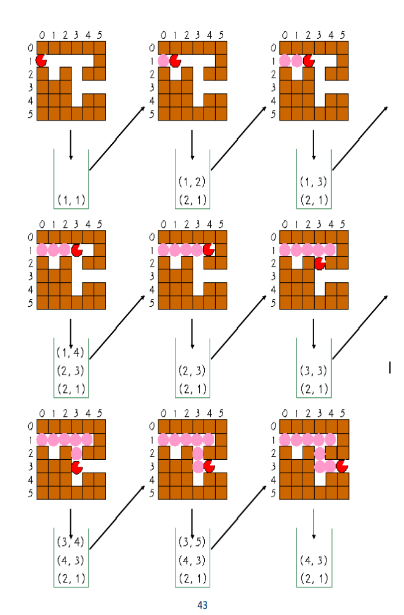
: we can determine whether a pair of strings is correct or not using stack.

1. calculation of Formulas

by changing basic expression to Postfix expression, we calculate more easily using stack.



1. Maze Search Problem

-stores the possible directions in the current position on the stack

- when it reaches a dead end, it takes the next seek position out of the stack.

**Queue**

* FIFO (First In, First Out)
* Delete the front of the data, and insert operations are performed at the end.
* The time complexity for all operations is O(1)
* The location where the deletion takes place is called front, and the location where the insertion takes place is called rear.
* Insertion is enqueue, deletion is dequeue.

**Queue : Operation:**

Create() = Creates a queue.

Init (q) = Initialize the queue.

is\_empty (q) = Checks if the queue is empty.

is\_full (q) = Checks whether the queue is full.

enqueue (q, e) = Add an element at the rear of the queue.

dequeue (q) = return the element at the front of the queue and delete it.

peek (q) = Returns the previous element without deleting it from the queue.

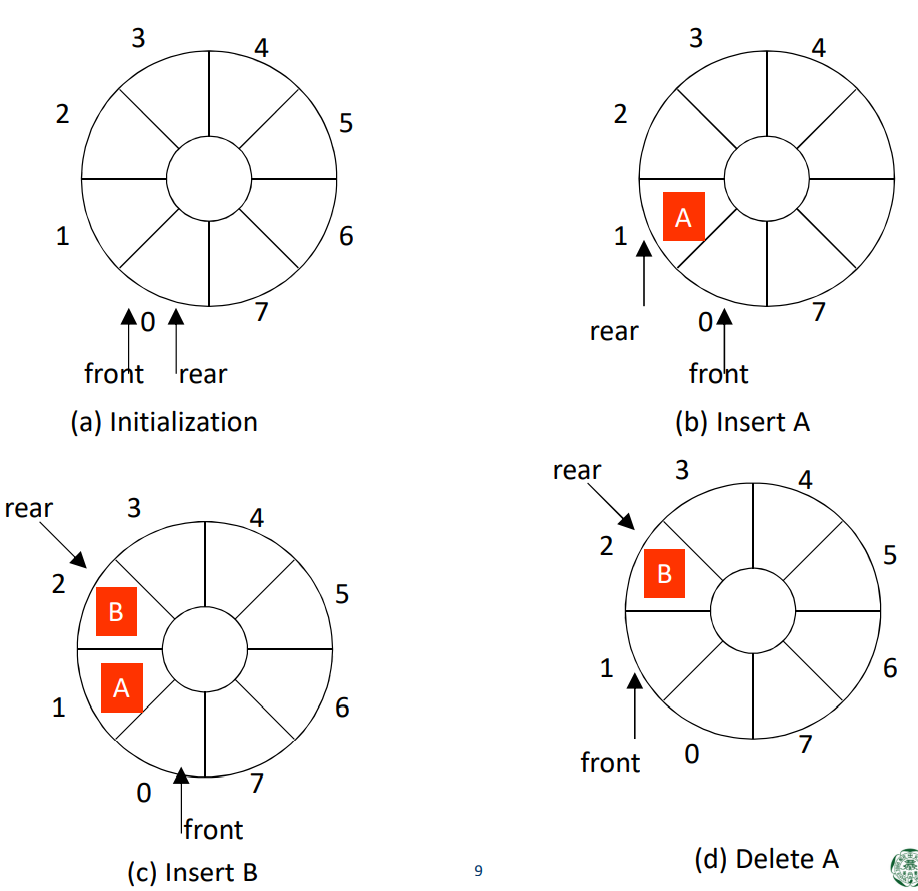
**Linear queue**

: A linear queue is a linear data structure that serves the request first, which has been arrived first. It consists of data elements which are connected in a linear fashion. It has two pointers, i.e., front and rear, where the insertion takes place from the front end, and deletion occurs from the front end. Implements a queue using arrays linearly

But, Linear queues require a process of pulling data forward.  
And a circular queue is used as a data structure to compensate for the problem of this linear queue!

**Circular queue**

**:** Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **‘Ring Buffer’**. In a normal Queue, we can insert elements until queue becomes full. But once queue becomes full, we can not insert the next element even if there is a space in front of queue.

****

‘Empty’: front == rear • ‘Full’: front% M == (rear + 1)% M