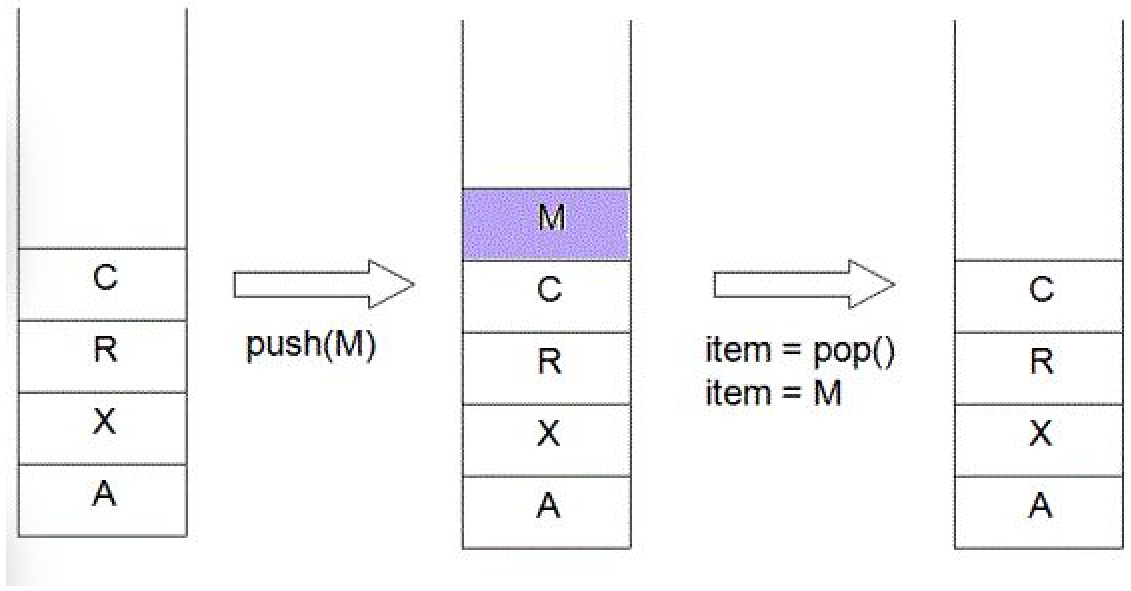
CSE 204

<u>Assignment 3</u>

Implementing Stack and Queue using Array

Task-1 (Stack Implementation using Array)

A stack is a container associated with objects which are inserted or removed based on the last-in first-out (LIFO) principle.

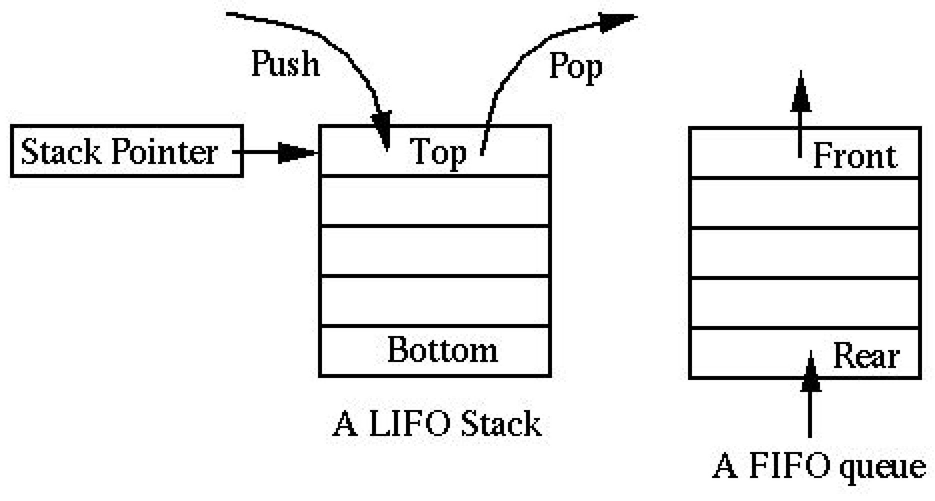


In this assignment, you need to design a stack using array. Use dynamic memory allocation to allocate memory for the array. Initially you can start with allocating memory for 10 elements. If at any point, stack size becomes equal to allocated memory, then allocate additional memory for 10 elements. The following standard operations must be supported:

<u>empty()</u> – Returns whether the stack is empty – Time Complexity : O(1)
<u>size()</u> – Returns the size of the stack – Time Complexity : O(1)
<u>top()</u> – Returns a reference to the top most element of the stack – Time Complexity : O(1)
<u>push(x)</u> – Adds the element ‘x’ at the top of the stack – Time Complexity : O(1)
<u>pop()</u> – Deletes the top most element of the stack – Time Complexity : O(1)

Task-2 (Queue Implementation using Array)

Queue is a data structure that maintain "**First In First Out**" (**FIFO**) order. Implement a queue using array. Maintain the memory of the array as the same way mentioned above.



The following standard operations must be supported:

empty() – Returns whether the queue is empty - Time complexity: O(1)

size() – Returns the size of the queue - Time complexity: O(1)

front() and rear()– **front()** function returns a reference to the first element of the queue. **rear()** function returns a reference to the last element of the queue - Time complexity: O(1)

enqueue(x) – adds the element 'x' at the end of the queue - Time complexity: O(1)

dequeue() – deletes the first element of the queue - Time complexity: O(1)