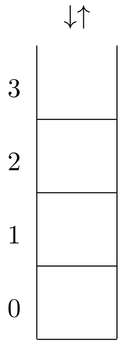
Stacks and Queues

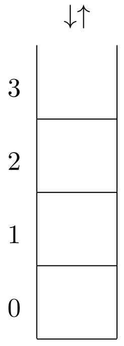
1. Assume a stack is implemented with a fixed size **array** as shown below. Also assume that the `top` pointer is initialized to `-1`. Show the state of the stack after the following operations. Be sure to indicate the final location of the `top` pointer.

   a) **push(3), push(2), push(1)**



   Continuing from where you left off above, do the remaining operations in the stack below
   (the **BOLD** operations are the remaining operations). Don't forget to label the location of the `top` pointer.

   b) push(3), push(2), push(1), **pop(), push(4), pop(), pop(), push(5)**



2. The stack described in question #1 initializes the `top` pointer to `-1`. Write pseudocode for an `isEmpty` method that utilizes the `top` pointer to determine if the stack is empty. Your method should return `true` if the stack is empty, `false` otherwise.

3. Assume a queue is implemented with a fixed size **array** as shown below. Also assume that both the `head` and `tail` pointers are initialized to 0. The first enqueue(3) operation writes the value 3 to index 0 of the backing array. Provide the final configuration of the backing array after the sequence of operations has completed. Also, indicate the position of both the `head` and `tail` pointers.

a) **enqueue(3), enqueue(2), enqueue(1)**

0	1	2	3

Continuing from where you left off above, do the remaining operations in the queue below (the **BOLD** operations are the remaining operations). Don't forget to label the location of the `head` and `tail` pointers.

b) enqueue(3), enqueue(2), enqueue(1), **dequeue(), enqueue(4), dequeue(), enqueue(5), dequeue()**

0	1	2	3

4. The queue described in question #2 initializes both the `head` and `tail` pointers to 0. Write pseudocode for an `isEmpty` method that utilizes the `head` and `tail` pointers to determine if the queue is empty. Your method should return `true` if the queue is empty, `false` otherwise.

5. List one advantage and one disadvantage of using a *preallocated*, i.e. fixed size, backing array to implement stacks and queues.