Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

Implement the MyStack class:

* void push(int x) Pushes element x to the top of the stack.
* int pop() Removes the element on the top of the stack and returns it.
* int top() Returns the element on the top of the stack.
* boolean empty() Returns true if the stack is empty, false otherwise.

**Notes:**

* You must use **only** standard operations of a queue, which means that only push to back, peek/pop from front, size and is empty operations are valid.
* Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations.

**Example**

class MyStack {

Queue<Integer> q=new LinkedList<Integer>();

Queue<Integer> q1=new LinkedList<Integer>();

public MyStack() {

}

public void push(int x) {

q1.add(x);

while(!q.isEmpty())

{

q1.add(q.peek());

q.remove();

}

Queue<Integer> temp=q;

q=q1;

q1=temp;

}

public int pop() {

if(q.isEmpty())

return-1;

int x=q.peek();

q.remove();

return x;

}

public int top() {

if(q.isEmpty())

return -1;

return q.peek();

}

public boolean empty() {

if(q.isEmpty())

return true;

return false;

}

}

/\*\*

\* Your MyStack object will be instantiated and called as such:

\* MyStack obj = new MyStack();

\* obj.push(x);

\* int param\_2 = obj.pop();

\* int param\_3 = obj.top();

\* boolean param\_4 = obj.empty();

\*/

