**Q1. Queue using two Stacks**

Implement a Queue using 2 stacks s1 and s2 .
A Query Q is of 2 Types
(i) 1 x (a query of this type means pushing 'x' into the queue)
(ii) 2 (a query of this type means to pop element from queue and print the poped element)

Note :- If there is no element return -1 as answer while popping.

Example 1:

Input:
5
1 2 1 3 2 1 4 2

Output:
2 3

Explanation:
In the first test case
1 2 the queue will be {2}
1 3 the queue will be {2 3}
2 popped element will be 2 the queue
will be {3}
1 4 the queue will be {3 4}
2 popped element will be 3.
Example 2:

Input:
4
1 2 2 2 1 4

Output:
2 -1

Explanation:
In the second test case
1 2 the queue will be {2}
2 popped element will be 2 and
then the queue will be empty
2 the queue is empty and hence -1
1 4 the queue will be {4}.

Expected Time Complexity : O(1) for push() and O(N) for pop() or O(N) for push() and O(1) for pop()
Expected Auxiliary Space : O(1).

Constraints:
1 <= Q <= 100
1 <= x <= 100

**Q2. Queue Reversal**
Given a Queue Q containing N elements. The task is to reverse the Queue. Your task is to complete the function rev(), that reverses the N elements of the queue.

Example 1:

Input:
6
4 3 1 10 2 6
Output:
6 2 10 1 3 4
Explanation:
After reversing the given elements of the queue , the resultant queue will be 6 2 10 1 3 4.
Example 2:

Input:
4
4 3 2 1
Output:
1 2 3 4
Explanation:
After reversing the given elements of the queue , the resultant queue will be 1 2 3 4.
Expected Time Complexity : O(n)
Expected Auxiliary Space : O(n)

Constraints:
$1 \leq N \leq 105$
$1 \leq$ elements of Queue $\leq 105$

**Q3. Reverse First K elements of Queue**
Given an integer K and a queue of integers, we need to reverse the order of the first K elements of the queue, leaving the other elements in the same relative order.

Only following standard operations are allowed on queue.

enqueue(x) : Add an item x to rear of queue
dequeue() : Remove an item from front of queue
size() : Returns number of elements in queue.
front() : Finds front item.
Note: The above operations represent the general processings. In-built functions of the respective languages can be used to solve the problem.
Example 1:

Input:
5 3
1 2 3 4 5

Output:
3 2 1 4 5

Explanation:
After reversing the given
input from the 3rd position the resultant
output will be 3 2 1 4 5.

Example 2:

Input:
4 4
4 3 2 1

Output:
1 2 3 4

Explanation:
After reversing the given
input from the 4th position the resultant
output will be 1 2 3 4.
Expected Time Complexity : O(N)
Expected Auxiliary Space : O(K)

Constraints:
1 <= N <= 1000
1 <= K <= N

**Q4. Stack using two queues**
Implement a Stack using two queues q1 and q2.

Example 1:

Input:
push(2)
push(3)

pop()
push(4)
pop()
Output: 3 4
Explanation:
push(2) the stack will be {2}
push(3) the stack will be {2 3}
pop()  poped element will be 3 the
stack will be {2}
push(4) the stack will be {2 4}
pop()  poped element will be 4
Example 2:

Input:
push(2)
pop()
pop()
push(3)
Output: 2 -1

Expected Time Complexity: O(1) for push() and O(N) for pop() (or vice-versa).
Expected Auxiliary Space: O(1) for both push() and pop().

Constraints:
1 <= Number of queries <= 100
1 <= values of the stack <= 100

**Q5. Queue Operations**
Given N integers, the task is to insert those elements in the queue. Also, given M integers, the task is to find the frequency of each number in the Queue.
Note:

insert() will be called N times by main().
findFrequency() will be called M times by main();
Where k is each element passing through respective function calls.

Example 1:

Input:
N = 8
1 2 3 4 5 2 3 1
M = 5
1 3 2 9 10
Output:
2
2
2
-1
-1
Explanation:
After inserting 1, 2, 3, 4, 5, 2, 3 and 1 into the queue, frequency of 1 is 2, 3 is 2 and 2 is 2. Since 9 and 10 are not there in the queue we output -1 for them.
Example 2:

Input:
N = 6
1 2 1 1 1 4
M = 4
1 5 4 3
Output:
4
-1
1
-1
Explanation:
After inserting 1, 2, 1, 1, 1 and 4 into the queue, frequency of 1 is 4 and that of 4 is 1. Since 5 and 3 are not there in the queue we output -1 for them.

Expected Time Complexity: O(N*M)
Expected Space Complexity: O(N)

Constraints:
1 <= N <= 10^3
1 <= M <= 10^3
1 <= Elements of Queue <= 10^6