2nd Jan 2023

FIRST :-

Stack using two queues Easy

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
```

Your Task:

Since this is a function problem, you don't need to take inputs. You are required to complete the two methods **push()** which takes an integer **'x'** as input denoting the element to be pushed into the stack and **pop()** which returns the integer poped out from the stack(**-1** if the stack is empty).

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

CODE SECTION:-

```
/* The structure of the class is
class QueueStack{
private:
    queue<int> q1;
    queue<int> q2;
public:
    void push(int);
    int pop();
};
 */

// Function to push an element into stack using two queues.
void QueueStack ::push(int x)
{
    // Your Code
    q2.push(x);
    while (!q1.empty())
    {

        q2.push(q1.front());
        q1.pop();
    }
    swap(q1, q2);
}
```

```cpp
// Function to pop an element from stack using two queues.
int QueueStack ::pop()
{
    // Your Code

    if (q1.empty())
    {
        return -1;
    }

    int x = q1.front();
    q1.pop();

    return x;
}
```

SECOND QUESTION:-

Queue using stack : Easy

Implement a Queue using two stack **s1** and **s2**.

Example 1:

```
Input:
enqueue(2)
enqueue(3)
dequeue()
enqueue(4)
dequeue()
Output: 2 3
Explanation:
enqueue(2) the queue will be {2}
enqueue(3) the queue will be {3 2}
dequeue() the poped element will be 2
the stack will be {3}
enqueue(4) the stack will be {4 3}
dequeue() the poped element will be 3.
```

Example 2:

```
Input:
enqueue(2)
dequeue()
dequeue()
Output: 2 -1
```

Your Task:

Since this is a function problem, you don't need to take inputs. You are required to complete the two methods **enqueue()** which takes an integer **'x'** as input denoting the element to be pushed into the queue and **dequeue()** which returns the integer poped out from the queue.

Expected Time Complexity: O(1) for **enqueue()** and O(n) for **dequeue()**
Expected Auxiliary Space: O(1) for both **enqueue()** and **dequeue()**

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

CODE SECTION:-

```cpp
// User function Template for C++
class Queue
{
    stack<int> s1, s2;
public:
    // INDEED revise
    void enqueue(int x)
    {
        while (!s1.empty())
        {
            s2.push(s1.top());
            s1.pop();
        }
        s1.push(x);
        while (!s2.empty())
        {   s1.push(s2.top());
            s2.pop();
        }
    }
    int dequeue()
    {

        if (s1.empty())
        {
            return -1;
        }

        int x = s1.top();
        s1.pop();
        return x;
    }
};
```

-: DONE FOR TODAY:-