Implement a stack using single queue

We are given queue data structure, the task is to implement stack using only given queue data structure.

We have discussed a solution that uses two queues. In this article, a new solution is discussed that uses only one queue. This solution assumes that we can find size of queue at any point. The idea is to keep newly inserted element always at front, keeping order of previous elements same. Below are complete steps.

```
// x is the element to be pushed and s is stack
push(s, x)
  1) Let size of q be s.
  1) Enqueue x to q
  2) One by one Dequeue s items from queue and enqueue them.

// Removes an item from stack
pop(s)
  1) Dequeue an item from q
```

Below is C++ implementation of the idea.

```
// C++ program to implement a stack using
// single queue
#include<bits/stdc++.h>
using namespace std;

// User defined stack that uses a queue
class Stack
{
    queue<int>q;
public:
    void push(int val);
    void pop();
    int top();
    bool empty();
};

// Push operation
void Stack::push(int val)
{
    //  Get previous size of queue
    int s = q.size();

    // Push current element
    q.push(val);

    // Pop (or Dequeue) all previous
    // elements and put them after current
    // element
    for (int i=0; i<s; i++)
    {
        // this will add front element into
        // rear of queue
        q.push(q.front());

        // this will delete front element
        q.pop();
    }
}

// Removes the top element
void Stack::pop()
{
    if (q.empty())
```

```
        cout << "No elements\n";
    else
        q.pop();
}

// Returns top of stack
int  Stack::top()
{
    return (q.empty())? -1 : q.front();
}

// Returns true if Stack is empty else false
bool Stack::empty()
{
    return (q.empty());
}

// Driver code
int main()
{
    Stack s;
    s.push(10);
    s.push(20);
    cout << s.top() << endl;
    s.pop();
    s.push(30);
    s.pop();
    cout << s.top() << endl;
    return 0;
}
```

Output :

```
20
10
```