HackerRank will be undergoing a scheduled maintenance on Monday May 29, from 5 pm GMT to 10 pm GMT. The site will be unavailable at this time.

Queues: A Tale of Two Stacks

by saikiran9194

Check out the resources on the page's right side to learn more about queues. The video tutorial is by Gayle Laakmann McDowell, author of the best-selling interview book Cracking the Coding Interview.

A queue is an abstract data type that maintains the order in which elements were added to it, allowing the oldest elements to be removed from the front and new elements to be added to the rear. This is called a *First-In-First-Out* (FIFO) data structure because the first element added to the queue (i.e., the one that has been waiting the longest) is always the first one to be removed.

A basic queue has the following operations:

- *Enqueue*: add a new element to the end of the queue.
- *Dequeue*: remove the element from the front of the queue and return it.

In this challenge, you must first implement a queue using *two stacks*. Then process q queries, where each query is one of the following 3 types:

1. `1 x`: Enqueue element x into the end of the queue.
2. `2`: Dequeue the element at the front of the queue.
3. `3`: Print the element at the front of the queue.

Input Format

The first line contains a single integer, q, denoting the number of queries.
Each line i of the q subsequent lines contains a single query in the form described in the problem statement above. All three queries start with an integer denoting the query $type$, but only query 1 is followed by an additional space-separated value, x, denoting the value to be enqueued.

Constraints

- $1 \leq q \leq 10^5$
- $1 \leq type \leq 3$
- $1 \leq |x| \leq 10^9$
- It is guaranteed that a valid answer always exists for each query of type 3.

Output Format

For each query of type 3, print the value of the element at the front of the queue on a new line.

Sample Input

```
10
1 42
2
```

```
1 14
3
1 28
3
1 60
1 78
2
2
```

Sample Output

```
14
14
```

Explanation

We perform the following sequence of actions:

1. Enqueue 42; $queue = \{42\}$.
2. Dequeue the value at the head of the queue, 42; $queue = \{\}$.
3. Enqueue 14; $queue = \{14\}$.
4. Print the value at the head of the queue, 14; $queue = \{14\}$.
5. Enqueue 28; $queue = \{14 \leftarrow 28\}$.
6. Print the value at the head of the queue, 14; $queue = \{14 \leftarrow 28\}$.
7. Enqueue 60; $queue = \{14 \leftarrow 28 \leftarrow 60\}$.
8. Enqueue 78; $queue = \{14 \leftarrow 28 \leftarrow 60 \leftarrow 78\}$.
9. Dequeue the value at the head of the queue, 14; $queue = \{28 \leftarrow 60 \leftarrow 78\}$.
10. Dequeue the value at the head of the queue, 28; $queue = \{60 \leftarrow 78\}$.

Submissions: 16850
Max Score: 30
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

Need Help?

5:46

Queues
Stacks

More

Current Buffer (saved locally, editable)

Python 2

```
class MyQueue(object):
    def __init__(self):
        self.first = []
        self.second = []

    def peek(self):


    def pop(self):


    def put(self, value):


queue = MyQueue()
t = int(raw_input())
for line in xrange(t):
    values = map(int, raw_input().split())

    if values[0] == 1:
        queue.put(values[1])
    elif values[0] == 2:
        queue.pop()
    else:
        print queue.peek()


```

Line: 1 Col: 1

Upload Code as File ☐ **Test against custom input** Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature