

You have a `RecentCounter` class which counts the number of recent requests within a certain time frame.

Implement the `RecentCounter` class:

- `RecentCounter()` Initializes the counter with zero recent requests.
- `int ping(int t)` Adds a new request at time `t`, where `t` represents some time in milliseconds, and returns the number of requests that has happened in the past `3000` milliseconds (including the new request). Specifically, return the number of requests that have happened in the inclusive range `[t - 3000, t]`.

It is **guaranteed** that every call to `ping` uses a strictly larger value of `t` than the previous call.

Example 1:

Input

```
["RecentCounter", "ping", "ping", "ping", "ping"]  
[[], [1], [100], [3001], [3002]]
```

Output

```
[null, 1, 2, 3, 3]
```

Explanation

```
RecentCounter recentCounter = new RecentCounter();  
recentCounter.ping(1);      // requests = [1], range is [-2999,1],  
return 1  
recentCounter.ping(100);    // requests = [1, 100], range is  
[-2900,100], return 2  
recentCounter.ping(3001);   // requests = [1, 100, 3001], range is  
[1,3001], return 3  
recentCounter.ping(3002);   // requests = [1, 100, 3001, 3002],  
range is [2,3002], return 3
```

```

class RecentCounter {
    queue<int> requests;
public:
    RecentCounter() {

    }

    int ping(int t) {
        // pushing request in queue
        requests.push(t);
        // determining start time
        int startTime = t - 3000;

        while (!requests.empty() && requests.front() < startTime)
            requests.pop();

        return requests.size();
    }
};

/**
 * Your RecentCounter object will be instantiated and called as such:
 * RecentCounter* obj = new RecentCounter();
 * int param_1 = obj->ping(t);
 */

```