## Logger Rate Limiter

Design a logger system that receive stream of messages along with its timestamps, each message should be printed if and only if it is **not printed in the last 10 seconds**.

Given a message and a timestamp (in seconds granularity), return true if the message should be printed in the given timestamp, otherwise returns false.

It is possible that several messages arrive roughly at the same time.

**Example:**

```
Logger logger = new Logger();

// logging string "foo" at timestamp 1
logger.shouldPrintMessage(1, "foo"); returns true;

// logging string "bar" at timestamp 2
logger.shouldPrintMessage(2,"bar"); returns true;

// logging string "foo" at timestamp 3
logger.shouldPrintMessage(3,"foo"); returns false;

// logging string "bar" at timestamp 8
logger.shouldPrintMessage(8,"bar"); returns false;

// logging string "foo" at timestamp 10
logger.shouldPrintMessage(10,"foo"); returns false;

// logging string "foo" at timestamp 11
logger.shouldPrintMessage(11,"foo"); returns true;
```

**Credits:**
Special thanks to @memoryless for adding this problem and creating all test cases.

## Solution 1

Instead of logging print times, I store when it's ok for a message to be printed again. Should be slightly faster, because I don't always have to add or subtract (e.g., `timestamp < log[message] + 10`) but only do in the `true` case. Also, it leads to a shorter/simpler longest line of code. Finally, C++ has 0 as default, so I can just use `ok[message]`.

---

### C++

```
class Logger {
public:

    map<string, int> ok;

    bool shouldPrintMessage(int timestamp, string message) {
        if (timestamp < ok[message])
            return false;
        ok[message] = timestamp + 10;
        return true;
    }
};
```

---

### Python

```
class Logger(object):

    def __init__(self):
        self.ok = {}

    def shouldPrintMessage(self, timestamp, message):
        if timestamp < self.ok.get(message, 0):
            return False
        self.ok[message] = timestamp + 10
        return True
```

---

### Java

```java
public class Logger {

    private Map<String, Integer> ok = new HashMap<>();

    public boolean shouldPrintMessage(int timestamp, String message) {
        if (timestamp < ok.getOrDefault(message, 0))
            return false;
        ok.put(message, timestamp + 10);
        return true;
    }
}
```

written by StefanPochmann original link here

## Solution 2

```
import java.util.concurrent.*;

public class Logger {
    ConcurrentHashMap<String, Integer> lastPrintTime;

    /** Initialize your data structure here. */
    public Logger() {
        lastPrintTime = new ConcurrentHashMap<String, Integer>();
    }

    /** Returns true if the message should be printed in the given timestamp, oth
erwise returns false. The timestamp is in seconds granularity. */
    public boolean shouldPrintMessage(int timestamp, String message) {
        Integer last = lastPrintTime.get(message);

        return last == null && lastPrintTime.putIfAbsent(message, timestamp) == n
ull
                || last != null && timestamp - last >= 10 && lastPrintTime.replac
e(message, last, timestamp);

    }
}
```

written by lop original link here

## Solution 3

```
 public class Logger {
HashMap<String,Integer> map;
/** Initialize your data structure here. */
public Logger() {
    map=new HashMap<>();
}

/** Returns true if the message should be printed in the given timestamp, otherwis
e returns false. The timestamp is in seconds granularity. */
public boolean shouldPrintMessage(int timestamp, String message) {
//update timestamp of the message if the message is coming in for the first time,o
r the last coming time is earlier than 10 seconds from now
    if(!map.containsKey(message)||timestamp-map.get(message)>=10){
        map.put(message,timestamp);
        return true;
    }
    return false;
}
```

}

written by bxiao0801 original link here