Problem 359: Logger Rate Limiter

Problem Information

Difficulty: Easy
Acceptance Rate: 76.72%
Paid Only: Yes
Tags: Hash Table, Design, Data Stream

Problem Description

Design a logger system that receives a stream of messages along with their timestamps. Each **unique** message should only be printed **at most every 10 seconds** (i.e. a message printed at timestamp `t` will prevent other identical messages from being printed until timestamp `t + 10`).

All messages will come in chronological order. Several messages may arrive at the same timestamp.

Implement the `Logger` class:

* `Logger()` Initializes the `logger` object. * `bool shouldPrintMessage(int timestamp, string message)` Returns `true` if the `message` should be printed in the given `timestamp`, otherwise returns `false`.

Example 1:

Input ["Logger", "shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage", "shouldPrintMessage"] [[], [1, "foo"], [2, "bar"], [3, "foo"], [8, "bar"], [10, "foo"], [11, "foo"]] **Output** [null, true, true, false, false, false, true] **Explanation** Logger logger = new Logger(); logger.shouldPrintMessage(1, "foo"); // return true, next allowed timestamp for "foo" is 1 + 10 = 11 logger.shouldPrintMessage(2, "bar"); // return true, next allowed timestamp for "bar" is 2 + 10 = 12 logger.shouldPrintMessage(3, "foo"); // 3 < 11, return false logger.shouldPrintMessage(8, "bar"); // 8 < 12, return false logger.shouldPrintMessage(10, "foo"); // 10 < 11, return false logger.shouldPrintMessage(11, "foo"); // 11 >= 11, return true, next allowed timestamp for "foo" is 11 + 10 = 21

Constraints:

* `0 <= timestamp <= 109` * Every `timestamp` will be passed in non-decreasing order (chronological order). * `1 <= message.length <= 30` * At most `104` calls will be made to `shouldPrintMessage`.

Code Snippets

C++:

```
class Logger {
public:
Logger() {

}

bool shouldPrintMessage(int timestamp, string message) {

}
};

/**
* Your Logger object will be instantiated and called as such:
* Logger* obj = new Logger();
* bool param_1 = obj->shouldPrintMessage(timestamp,message);
*/
```

Java:

```
class Logger {

public Logger() {

}

public boolean shouldPrintMessage(int timestamp, String message) {

}
}

/**
```

```
* Your Logger object will be instantiated and called as such:
* Logger obj = new Logger();
* boolean param_1 = obj.shouldPrintMessage(timestamp,message);
*/
```

Python3:

```
class Logger:

def __init__(self):


def shouldPrintMessage(self, timestamp: int, message: str) -> bool:



# Your Logger object will be instantiated and called as such:
# obj = Logger()
# param_1 = obj.shouldPrintMessage(timestamp,message)
```