HW 09: Catching Hackers

A web server is logging all the attempts to login. Associated with every login attempt is the IP address of the computer making the attempt. The log files have one line per attempt and have the following info: the time of the attempt, the IP address, and the response (SUCCESS or FAIL). You are tasked with trying to identify if there are malicious parties attempting to guess passwords. The first step is to identify suspicious behavior. We will decide that something like 3 failed logins within 5 seconds is suspicious and we will start by reporting these IP addresses. More generally, we will look for `k` consecutive failed logins within `s` seconds, where `k` and `s` are parameters. By "within `s` seconds", we mean that the difference in the time between the first and the last is less than `s`.

The input format

The log file will be formatted to have the timestamp, the IP address, and the word `SUCCESS` or `FAIL`, each contained in brackets. The file will be sorted by increasing time stamp. Here is a sample input file.

```
[1521482277.017901][192.168.1.1][FAIL]
[1521482923.605063][192.168.1.1][FAIL]
[1521482938.320314][1.1.1.1][SUCCESS]
[1521483145.456712][192.168.1.1][SUCCESS]
```

You have been provided with a `LogEntry` class that parses such strings.

The BadLoginDetector class

Write a class called `BadLoginDetector` and save it in a file called `badlogindetector.py`. It should be initialized with two arguments `k` and `s`. It should support two main operations:

- `__init__(self, k, s)` - initialize a new `BadLoginDetector` with threshold `k` for consecutive failures and time limit `s` seconds.
- `process(self, logentry)` - This takes a new logentry and returns `False` if this is the kth consecutive failed login from this IP address within the last s seconds. Otherwise, it returns `True`.

- `report(self)` - Return a sorted list of IP addresses from which k consecutive failed logins were attempted within s seconds. This assumes all the log entries were already processed.

The key to making a fast solution here is to use a Mapping data structure. The `LastK` data structure from the lab will come in handy.

Do not use a python dictionary. You can base your implementation on the code from the book, but you should expect to really understand it in order to get it to work.

Working with time stamps

The timestamps will be in seconds. This is a standard representation of time used in many computer systems. The idea is to fix a particular moment in time and then use the number of seconds since that moment to measure time.

Hints

Don't read these unless you feel you really need them.

You will want to store a mapping in which the keys are IP addresses and the values are `LastK` data structures storing the LastK login attempts. When a new log entry comes along, if it's a success, then you can clear the `LastK`. If it's a failure, you can add it to the LastK. If there are at least k items in there and the time difference between the first and last is less than s, then you have found a new bad login.