5694. Design Authentication Manager

My Submissions (/contest/biweekly-contest-48/problems/design-authentication-manager/submissions/) Back to Contest (/contest/biweekly-contest-48/)

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Medium

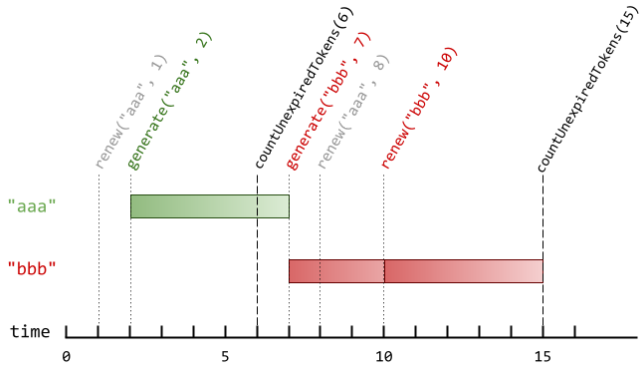
There is an authentication system that works with authentication tokens. For each session, the user will receive a new authentication token that will expire `timeToLive` seconds after the `currentTime` . If the token is renewed, the expiry time will be **extended** to expire `timeToLive` seconds after the (potentially different) `currentTime` .

Implement the `AuthenticationManager` class:

- `AuthenticationManager(int timeToLive)` constructs the `AuthenticationManager` and sets the `timeToLive` .
- `generate(string tokenId, int currentTime)` generates a new token with the given `tokenId` at the given `currentTime` in seconds.
- `renew(string tokenId, int currentTime)` renews the **unexpired** token with the given `tokenId` at the given `currentTime` in seconds. If there are no unexpired tokens with the given `tokenId` , the request is ignored, and nothing happens.
- `countUnexpiredTokens(int currentTime)` returns the number of **unexpired** tokens at the given currentTime.

Note that if a token expires at time `t` , and another action happens on time `t` (`renew` or `countUnexpiredTokens`), the expiration takes place **before** the other actions.

Example 1:




```
Input
["AuthenticationManager", "renew", "generate", "countUnexpiredTokens", "generate", "renew", "renew", "countUnexpiredTokens"]
[[5], ["aaa", 1], ["aaa", 2], [6], ["bbb", 7], ["aaa", 8], ["bbb", 10], [15]]
Output
[null, null, null, 1, null, null, null, 0]

Explanation
AuthenticationManager authenticationManager = new AuthenticationManager(5); // Constructs the AuthenticationManager with time
ToLive = 5 seconds.
authenticationManager.renew("aaa", 1); // No token exists with tokenId "aaa" at time 1, so nothing happens.
authenticationManager.generate("aaa", 2); // Generates a new token with tokenId "aaa" at time 2.
authenticationManager.countUnexpiredTokens(6); // The token with tokenId "aaa" is the only unexpired one at time 6, so return 1
authenticationManager.generate("bbb", 7); // Generates a new token with tokenId "bbb" at time 7.
authenticationManager.renew("aaa", 8); // The token with tokenId "aaa" expired at time 7, and 8 >= 7, so at time 8 the renew re
authenticationManager.renew("bbb", 10); // The token with tokenId "bbb" is unexpired at time 10, so the renew request is fulfil
authenticationManager.countUnexpiredTokens(15); // The token with tokenId "bbb" expires at time 15, and the token with tokenId
```

Constraints:

- `1 <= timeToLive <= 10^8`
- `1 <= currentTime <= 10^8`
- `1 <= tokenId.length <= 5`
- `tokenId` consists only of lowercase letters.
- All calls to `generate` will contain unique values of `tokenId` .
- The values of `currentTime` across all the function calls will be **strictly increasing**.
- At most `2000` calls will be made to all functions combined.

JavaScript

```javascript
/**
 * @param {number} timeToLive
 */
var AuthenticationManager = function(timeToLive) {

};

/**
 * @param {string} tokenId
 * @param {number} currentTime
 * @return {void}
 */
AuthenticationManager.prototype.generate = function(tokenId, currentTime) {

};

/**
 * @param {string} tokenId
 * @param {number} currentTime
 * @return {void}
 */
AuthenticationManager.prototype.renew = function(tokenId, currentTime) {

};

/**
 * @param {number} currentTime
 * @return {number}
 */
AuthenticationManager.prototype.countUnexpiredTokens = function(currentTime) {

};

/**
 * Your AuthenticationManager object will be instantiated and called as such:
 * var obj = new AuthenticationManager(timeToLive)
 * obj.generate(tokenId,currentTime)
 * obj.renew(tokenId,currentTime)
 * var param_3 = obj.countUnexpiredTokens(currentTime)
 */
```

Custom Testcase Use Example Testcases

Run Submit