CrowdTwist Design and Coding Test (Senior Software Engineer)

CrowdTwist Design and Coding Test

Welcome to the CrowdTwist Design and Coding Test!

Please read the following instructions carefully before beginning the test.

1. Solutions are judged primarily on correctness, but design, clarity, efficiency, and documentation are all taken into account.
2. All design and coding exercises are to be written as if you were on the job. Document as if you were working on the CrowdTwist codebase. Unit test as if you were working on the CrowdTwist codebase. Code as if you were working on the CrowdTwist codebase, and so on. If you write the coding exercise on the back of a napkin and take a picture for your submission, it had better be good!
3. Coding exercises can be written in any language. But if you are familiar with OO languages like PHP, Java, C#, C++, etc, please use that.
4. Coding exercises must use standard, operating system neutral libraries for the programming language.
5. Provide solutions as a .zip, .tar, or .tar.gz.
6. Document all assumptions. If you are uncertain what is being asked, use your best judgement and document your assumptions.

System Design Exercise

As part of a trading system, we need to design a subsystem which receives, manages and retrieves trading system logs. Every time a transaction occurs in the market, the trading system will send a TCP request to our subsystem, containing log information for the transaction that just occurred. The transaction information includes, but is not limited to, date, time, stock symbol and price.

How would you design this system? Where would you store the logs and how? How would you retrieve the logs? What are the pros and cons to your approach? Tradeoffs? What data structures would you use and why?

System Requirements

- Users of the trading system need to retrieve all transactions that happened in the system for a given time or time range, with seconds precision. For example, a user may make a request to our system for all transactions that occurred January 1st, 2013 00:00:00 for which they will expect all of the log information for every transaction that occurred at that second in time.
- No off-the-shelf database technologies (i.e. no Oracle, MySQL, DB2, SQL Server, MongoDB, Redis, etc).

Couple of things:

- There is no right or wrong answer. This is a critical thinking, problem solving, and technical design expertise exercise. We're interested to see what you come up with.
- A macro-level overview of the system you would implement is sufficient. This is a complex problem and we are by no means expecting detailed class diagrams, psuedo code, etc. Textual description of the system or diagrams with contextual descriptions are acceptable.
- There can be **a lot** of transactions in the system. It is very likely there will be N number of logs for a given second in time.
- The detailed log information isn't important (hint: except the date and time).
- This exercise requires no trading system expertise. A trading system is merely used as an example of a large, distributed system.

Coding Exercise

Consider a screen with single row of pixels. Pixels can be either Red or Yellow colors. We have to animate the pixels using the following rules :

1. Red pixels can only move to the right.
2. Yellow pixels can only move to the left.
3. Both color pixels will move at the same speed.
4. The pixels can pass through each other. So if the Red and Yellow colors occupy the SAME pixel then we have to display Orange color.

You will be given the initial conditions by a String init containing at each position a 'Y' for a leftward moving yellow pixel, an 'R' for a rightward moving Red pixel, or a '.' for an empty pixel. *init* shows all the positions on the screen. Initially, no location on the screen contains the two color pixels passing through each other.

We would like an animation of the process. At each unit of time, we want a string showing occupied locations with either a 'R' , 'Y' or 'O' (if the two

colors occupy same pixel) and unoccupied locations with a '.'. Create a class Animation that contains a method animate that is given an initial speed and a String *init* giving the initial conditions. The speed is the number of positions each particle moves in one time unit.

The method will return an array of strings in which each successive element shows the occupied locations at the next time unit. The first element of the return should show the occupied locations at the initial instant (at time = 0) in the 'Y','R', '.' format. The last element in the return should show the empty screen at the first time that it becomes empty.

Class: Animation

Method: animate(speed, init), where speed is an integer and init is a String.

You may assume the following constraints:
- speed will be between 1 and 10 inclusive
- *init* will contain between 1 and 50 characters inclusive
- each character in init will be '.' or 'Y', 'R' or 'O'

Examples

(Note that in the examples below, the double quotes should not be considered part of the input or output strings.)

0) 2, "..R...."

Returns:

```
{ "..R....",
  "....R..",
  "......R",
  "......." }
```

The single Red pixel starts at the 3rd position, moves to the 5th, then 7th, and then out of the screen.

1) 3, "RR..YRY"

Returns:

```
{ "RR..YRY",
  ".Y.OR..",
  "Y.....R",
  "......." }
```

At time 1, there are actually 5 colored pixels in the chamber, but two are passing through each other at the 4th position to give the Orange color.

2) 2, "YRYR.YRYR"

Returns:

```
{ "YRYR.YRYR",
  "Y..O.O..R",
  ".Y.Y.R.R.",
  ".Y.....R.",
  "........." }
```

At time 0 there are 8 colored pixels. At time 1, there are still 6 pixels, but only 4 positions are occupied since particles are passing through each other, giving Orange color.

3) 10, "RYRYRYRYRY"

Returns:

```
{ "RYRYRYRYRY",
  ".........." }
```

These pixels are moving so fast that they all exit the screen by time 1.

4) 1, "..."

Returns:

```
{ "..." }
```

5) 1, "YRRY.YR.YRR.R.YRRY."

Returns:

```
{ "YRRY.YR.YRR.R.YRRY.",
  "..ORY..O..RR.O..OR.",
  ".Y.OR.Y.R..RO.RY.RR",
  "Y.Y.RO...R.YRRYR..R",
  ".Y..YRR...O..OR.R..",
  "Y..Y..RR.Y.RY.RR.R.",
  "..Y....RO..YR..RR.R",
  ".Y.....YRRY..R..RR.",
  "Y.....Y..OR...R..RR",
  ".....Y..Y.RR...R..R",
  "....Y..Y...RR...R..",
  "...Y..Y.....RR...R.",
  "..Y..Y.......RR...R",
  ".Y..Y.........RR...",
  "Y..Y...........RR..",
  "..Y.............RR.",
  ".Y...............RR",
  "Y.................R",
  "..................." }
```