PAXOS

Software Engineering Coding Challenge

- This test is designed to help us evaluate a candidate's written code without imposing arbitrary time constraints on their work
- Candidates need to complete both challenges
- The response to the programming challenges can be implemented in your programming language of choice
- Here is something we wrote on why we think take-home tests are the best way for you to show off your skills: https://www.paxos.com/engineering/why-take-home-tests-are-awesome

How to Submit

- Send us your source code by either:
  - Create a private repo on github for your submission and add **paxos-recruiting** as a collaborator to your private repo
  - Download ZIP in github, and send us an email with it as an attachment or as a link to a cloud location (if you're compressing this file, please use .tar)

- Use the following naming convention: SOLUTION_FIRSTNAME_LASTNAME
- Please include in one or more readme files:
  - answers to the scaling/deployment questions [#1] and the big-O notation question [#2]
  - at least one example of how to run your code for each challenge
  - discussion of your approach [optional]
- Please include tests for your solutions.

PAXOS

Challenge #1 - Programming

Introduction

Create a small service deployed to a docker container that has two endpoints:

1. /messages takes a message (a string) as a POST and returns the SHA256 hash digest of that message (in hexadecimal format)
2. /messages/<hash> is a GET request that returns the original message. A request to a non-existent <hash> should return a 404 error.

Example

Let's say you expose port 8080:

```
$ curl -X POST -H "Content-Type: application/json" -d '{"message": "foo"}'
http://localhost:8080/messages

{

   "digest": "2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae"

}
```

You can calculate that your result is correct on the command line:

$ echo -n "foo" | shasum -a 256 2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae -

You can now query your service for the original message:

```
$ curl http://localhost:8080/messages/2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f9
8a5e886266e7ae

{

   "message": "foo"

}

$ curl -i http://localhost:8080/messages/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaa HTTP/1.0 404 NOT FOUND Content-Type: application/json Content-Length: 36
Server: Werkzeug/0.11.5 Python/3.5.1 Date: Wed, 31 Aug 2016 14:21:11 GMT

{

   "err_msg": "Message not found"

}
```

(your specifics may vary, all that matters is that you get a 404)

Scaling Question [required]

What would the bottleneck(s) be in your implementation as you acquire more users? How you might scale your microservice?

Deployment Question [required]

How would you improve your deployment process if you needed to maintain this application long term?

PAXOS

Challenge #2 - Programming

Introduction

You have been given a gift card that is about to expire and you want to buy gifts for 2 friends. You want to spend the whole gift card, or if that's not an option as close to the balance as possible. You have a list of sorted prices for a popular store that you know they both like to shop at. Your challenge is to find two distinct items in the list whose sum is minimally under (or equal to) the gift card balance.

The file contains two columns:

1. A unique identifier of the item. You can assume there are no duplicates.
2. The value of that item in cents. It is always a positive integer that represents the price in cents (1000 = $10.00).

Write a program to find the best two items. It takes two inputs:

1. A filename with a list of sorted prices
2. The balance of your gift card

If no two items have a sum that is less than or equal to the balance on the gift card, print "Not possible". You don't have to return every possible pair that is under the balance, just one such pair

Examples

```
$ cat prices.txt
Candy Bar, 500
Paperback Book, 700
Detergent, 1000
Headphones, 1400
Earmuffs, 2000
Bluetooth Stereo, 6000

$ find-pair prices.txt 2500
Candy Bar 500, Earmuffs 2000
```

```
$ find-pair prices.txt 2300
Paperback Book 700, Headphones
1400

$ find-pair prices.txt 10000
Earmuffs 2000, Bluetooth Stereo
6000

$ find-pair prices.txt 1100
Not possible
```

What is the big O notation for your program? [required]

Bonus Question [optional]

A. You are considering giving gifts to more people. Instead of choosing exactly 2 items, allow for 3 gifts.

B. How would you optimize your solution if you could not load the file in memory?