Programming Exercise

Maxwell Forest

Introduction

As part of the recruitment process at Maxwell Forest, we require you to write an application solving one of the problems below. It is intended to evaluate your ability to turn requirements into a working application.

Description

Solve **one** of the two problems on the next page. You should choose the problem most relevant to the position that you are applying for, however if you get stuck with one, attempt the other one instead. You should spend **around 2 hours** completing this exercise.

General requirements

- Unit tests are **mandatory** for all solutions
- Comment your code appropriately
- You are allowed to use libraries (unless otherwise stated), but not to solve the majority of the problem
- Dependency management might be a good idea (e.g. Gradle, CocoaPods)
- You are **not** allowed to use hybrid or cross-compile application frameworks (e.g. PhoneGap, Xamarin)
- Provide a README.md file describing how to run your code, what approach you took, what assumptions you made and any other relevant comments for the reviewer
- Make it as easy as possible for the reviewer to run your code, provide
  - IDE project, where the application can be run and tested, or
  - shell/batch scripts to run and test the application
  - make it clear in the README.md how the above items could be used
- The reviewer will be running OS X with Xcode and IntelliJ available, if your solution requires any additional packages to be installed, provide clear instructions

Submission

Upload your solution (source code only, no binaries) to a public repository (e.g. GitHub, Bitbucket) and email the link to careers@maxwellforest.com with the subject “Programming Exercise”.

Warning!

It is important that you do **not** use the company name anywhere in the project, including but not limited to the package structure and documentation (e.g. README.md file).

Problem #1: Apps

Software engineering requires a lot of caffeine so you decided to automate locating coffee with an app. Use the Foursquare API to find the closest coffee shops to your current location and display them in a list. The user experience is up to you: don't overdo it but make it look decent.

Requirements

- Application has to be native and written for **iOS (Objective-C, Swift)** or **Android (Java)**
- Current location has to update continuously (accuracy and frequency are up to you)
- Items must be sorted by distance in ascending order
- Each item has to show name, distance and address
- **Bonus:** Add some extra features to improve the user experience, such as ability to call the coffee shop or find it on the map (via Maps app)

Resources

- Foursquare API: https://developer.foursquare.com/docs/
- Client ID: ACAO2JPKM1MXHQJCK45IIFKRFR2ZVL0QASMCBCG5NPJQWF2G
- Client Secret: YZCKUYJ1WHUV2QICBXUBEILZI1DMPUIDP5SHV043O04FKBHL

Problem #2: Infrastructure

Your colleague working on the first problem needs you to develop a web service to integrate with. Your application will process multiple lists of coffee shops (CSV files) on launch and then listen on for search requests.

Requirements

- Application has to be written in **C/C++**, **Java**, **Ruby** or **Python**
- You are **not** allowed to use a web server, the application has to serve requests directly, however you **are** allowed to use libraries that can help
- In addition to the server application, you must provide a script with sample requests
- Use a file-based database for storage not requiring a DBMS installation (e.g. SQLite)
- Request/response formats are up to you but
  - requests must specify a location (lat/long) and/or a partial name, and
  - responses must specify all the fields ingested from the data sources
- Response has to contain the coffee shops with names filtered by the partial name (if provided) sorted by the distance from the provided location (if provided)
- **Bonus:** Make it an actual RESTful web service (HTTP requests and JSON responses)

Resources

- Data sources: *coffee_shops-1.csv*, *coffee_shops-2.csv*