Instruction to candidate

Introduction

In response to COVID-19 and the outbreak of the Delta variant, a quasi-government agency in Australia has engaged Versent to build and operate a data analytics platform to process CSV data collected by a website that gathers data generated by a mobile app to perform contact tracing. The *website* and *mobile-app* are being worked on by other companies that *doesn't* concern us.

Business background

- Upon installation of the mobile app, an individual's "phone number" is collected and verified.
- The mobile app will generate a new cryptographically random `tracking_id` (to create a new "tracking session").
- Whenever a person enters a location, the mobile app will automatically sign-in and sign-out. This represents a "visit".
- The mobile app's `tracking_id` changes over time (and generates a new "tracking session").
- A person becomes "infectous" (and therefore an "infecter"), capable of infecting others (who are "infectees") 24 hours (86,400 seconds) after they are "infected".
- A person stops being "infectous" after 14 days (exactly 1,209,600 seconds).
- A "close contact" is someone who were in the same location and at the same date/time -- as someone who was "infectous". This triggers an "infection event".
- For the sake of simplicity, an "infection event" is assumed to have occurred regardless of the actual distance between the "infecter" and "infectee". As long as both persons were in the same location within an overlapping time window. The "infection event" starts at a date/time when either:
  - An "infecter" begins a location visit (and the "infectee" is already present at the location); or
  - When an "infectee" begins a location visit (and the "infecter" is already present at the location).
- For simplicity, we can ignore the risk that COVID-19 viral particles hangs around in the air, thereby infecting new "infectees" after the "infecter" has left a "location".
- For simplicity, we can ignore the risk that COVID-19 viral particles might drift in air currents, thereby potentially infecting "infectees" in a different location *adjacent* to the "infecter's" location.
- For simplicity, we can assume no-one is infected unless otherwise stated explicitly.
- The website will provide you two CSV files (described in greater details below) along with a technical specifications of the fields in the CSV.

Input data - Phone Registration data

A file that represents a list of "tracking sessions". Each line represents a single "tracking session". Each session is uniquely identified by a `tracking_id`, and is associated with a person's "phone number" and the date/times that the "tracking session" started and ended. Therefore, a single person's phone, can have multiple "tracking sessions" (and hence `tracking_id`) over time.

The input file is a CSV (comma-separated values) file with the following columns (in order):

1. `tracking_id`
2. `session_datetime_start`
3. `session_datetime_end`
4. `phone_number`

A sample phone registration data file is provided at: `data/phone_registration.csv`

Input file - Visitation data

A file that represents a list of "visits" of people to locations. Each line represents a single visit of a person to a single location. Each visit is associated with a person (via the `tracking_id`), a location (via the `location_id`) and the date/times that a visit started/ended.

The input file is a CSV (comma-separated values) file with the following columns (in order):

1. `tracking_id`
2. `location_id`
3. `visit_datetime_start`
4. `visit_datetime_end`

A sample visitation data file is provided at: `data/visits.csv`

Technical specifications of fields

- All fields are case-sensitive unless otherwise specified (only UUID).
- A "regex" below refers to an "Extended Regular Expression":
  - See: https://en.wikipedia.org/wiki/Regular_expression#POSIX_extended
- All date/time fields are in the ISO8601 format of `YYYY-MM-DDThh:mm:ssZ`
  - e.g. `2021-07-12T08:51:26Z`
  - strftime format string of `%Y-%m-%dT%H:%M:%SZ`
  - The trailing `Z` denotes that the date/time is in the UTC (aka Zulu) timezone.
  - See: https://en.wikipedia.org/wiki/ISO8601
- All phone numbers are Australian local phone numbers (with a `tel:` prefix):
  - This means you won't encounter international, free-call, toll-free, premium rate or emergency numbers.
  - Valiation regex: `/tel:\+61[0-9]{9}/`
  - For example: `tel:+61401234567`, `tel:+61398765432`
- The `tracking_id` field is a UUID/GUID (see: https://en.wikipedia.org/wiki/Universally_unique_identifier) along with a `uuid:` prefix.
  - Validation regex: `/uuid:[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}/
  - For example: `uuid:68012f1b-32be-4a1b-9ba1-ef1c800b5b7f`
  - BTW - UUID are case-insensitive. So `uuid:aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa` and `uuid:AAAAAAAA-AAAA-AAAA-AAAA-AAAAAAAAAAAA` are identical.
- The `location_id` field is the string `dpid:` concatenated with a DPID (Delivery Point Identifier) from Australia Post's National Address File. It is a unique 8 digit ID.
  - Validation regex: `/dpid:[0-9]{8}/`
  - For example: `dpid:63121907`

Tasks

1. Design and implement a set of classes to read the CSV data into a well-organised, data-structure in memory.
   - Sample data is provided and should easily fit. But you may elect to generate your own data.
   - You may hard-code the filenames in this task.
2. Given a phone number and the date/time that the person (e.g. "person_a") was "infected" (but not yet "infectous"), identity all their (direct) close-contacts (e.g. "person_b"). This means that "person_a" might have infected "person_b".
   - This is forward contact tracing for a single generation.
   - Output the phone numbers of all close contacts please.
3. Extension of task 2. Assuming that "close contacts" are *always* "infected":
   - Identity the chain of forwards infections to 5 generations.
   - Reminder: people go from being "infected" to "infectous" 24 hours after "infection".
   - Output the phone numbers of all close contacts as well as the generation they were first infected.

4. Variation of task
5. In the previous tasks, there were no requirements on how the classes were executed (and therefore, it was at your option). In this task, please adapt your code to be a CLI, a GUI, or a web-API (your choice). Basically, in previous tasks, it was acceptable to require a recompile if the person infected (or their date of infection) changed.
6. Given the nature of a take-home test (and the lack of time), you aren't necessarily expected to cover all corner cases; or handle non-functional requirements. We recommend that you leave this last, but spend 10-30 minutes on this please.

   Please share:

   - A list of short-cuts that you took; or technical-debt; or limitations/bugs that is in your code.
   - Your recommendations for future work, or obvious extensions?
   - What would you do if you had much more time please?

Final notes

- This isn't a pass/fail test. This is an opportunity for you to demonstate your breadth/depth of your skills/expertise.
- Please complete the tasks in one (or more) programming language(s) and framework(s) of your choice.
- You do *not* have to complete every task.
  - This test is targeted at candidates with a wide variety of experiences and seniority.
  - Furthermore, different programming languages have different levels of overhead which we'll factor into our consideration.
- Nonetheless, we recommend that you read every tasks before beginning as that may factor into your design decisions.
- While you can complete tasks in any order (including skipping tasks), we recommend prioritizing tasks in the order given.
  - In some cases, a task extends a previous task.
  - In other cases (given the limited time candidates can devote), we want to ensure you don't waste time on certain aspects of development. For example, while we appreciate unit and integration tests, we value getting the tasks completed correctly over having 100% code coverage.
- Just do whatever you feel is reasonable with 6 hours work (though you may exceed that if you wish). If you cannot devote 6 hours of effort, we understand as candidates often have prior plans and personal responsibilities. Please devote however much effort you feel is reasonable and then share the rough amount of time you've spent as part of your submission please.
- You are absolutely allowed (and encouraged) to use books, Google and other resources on the Internet.
- Clean, readable, idiomatic and maintainable code would be appreciated please.
- Extensibility should be considered.
- A git commit-history would be preferred, with small changes committed often so we can see your approach.
- As in real-life, development tasks aren't always well-defined. Feel free to make any assumptions you feel is reasonable, but please identity and communicate these assumptions please.
- Please provide a README with documentation on requirements and building, testing and execution steps. You may wish to use it to explain any design decisions.
- Your code doesn't *have* to work. Pseudocode is an inferior, but acceptable alternative which might demonstrate your thought processes and design skills. Good, debugged, working code is best, but a good design will still get you points.
  - If your code works, you might be asked to demonstrate during the interview. Please bring your computer if possible.