# 2018 Career Fair Programming Competition
# Part A – Code Challenge

Problem Revealed: Sunday, 4 March 2017
Submissions due: **Sunday, 11 March 2017** at **10pm** (on Courseware)

**Content**

## Problem Description

The Ebola outbreak of 2014-2015 in West Africa, was the deadliest occurrence of the disease since it was first discovered in 1976. Over 28,000 cases were reported overall, and more than 11,000 people were reported to have died of the disease in six countries; Liberia, Guinea, Sierra Leone, Nigeria, the US and Mali. The last country, Liberia, was declared Ebola free in January 2016. There needs to be 42 days without any new cases for a country to be declared Ebola-free.

In this coding challenge, you will process some data related to the Ebola outbreak.

**Task 1**

You have been provided with a log file (in CSV format) of time-series data of the cumulative number of Ebola cases and deaths in a given country/region. You are to write a programme which determines answers to the following questions:

*(Difficulty Level: 1)*

a) When was the last case recorded?
b) When was the last death recorded?
c) Based on the data, on what date could the country be declared Ebola-free?
d) Between which dates was the infection rate the highest?
e) Between which was the death rate the highest?

*(Difficulty Level: 2)*

f) How many peaks were there in the infection rate over time, and on when did these occur?
g) How many peaks were there in the death rate over time, and when did these occur?

**Task 2** *(Difficulty Level: 3)*

You have been provided with a log file (in CSV format) containing time-series data of cumulative number of Ebola cases and deaths in several countries and localities. You have been given a second file with a list of time series data corresponding to a portion of one of the countries or localities in the Ebola log file. However, the data is not labelled so you do not know which country or region the data corresponds to, or what dates it corresponds to. Write a programme to determine which country & locality the data corresponds to, whether it's data related to cases or deaths, and what the starting date of the data is.

Note that, for both tasks, your program will be tested with different data from what has been provided to you. As such, it is important that you write your program such that it adheres to the guidelines described in this document, so that your code can be run with a different input file without re-compiling your code.

## Program Execution Specifications

Your programs can be written in either Python or Java. In either case, you should write your program such that it can be executed from the command-line, taking the filenames as command-line parameters.

For **task 1**, your programme should be run from the command line as follows:

```
<interpreter> <programme> <simple_ebola_data >
```

where:

| | |
|---|---|
| `<interpreter>` | is the name of the python interpreter or Java interpreter, |
| `<programme>` | is the name of your programme, and |
| `<simple_ebola_data>` | is the name of the file containing the time series data of Ebola cases and deaths for a single country/locality |

For **task 2**, your programme should be run from the command line as follows:

```
<interpreter> <programme_name> <complex_ebola_data> <partial_time_series>
```

where the first two parameters have the same meaning as above, and

| | |
|---|---|
| `<complex_ebola_data>` | is the name of the file containing the time series data of Ebola cases and deaths for several countries and localities, and |
| `<partial_time_series_file>` | contains the partial time series data that you are trying to match to the data in the `<complex_ebola_data>` file. |

The format of the input files are described in the next section.

Your programme should write its results, include the amount of time it took to answer each question, to an output file, whose format is described later in this document.

## Input File Formats

For tasks 1 and 2, the **simple ebola data** and **complex ebola data** files are CSV (comma-separate-value) files. Note that CSV files are simply plain text files in which the data values on each line are separated by commas. You can read such files in your Java or Python program like you would read any text file. You can view such files with a text editor (such as Notepad). You can also use Excel to view such files in a nice tabular format.

The **simple ebola data** and **complex ebola data** files are structured as follows:

- Line 1 has the column headers
- Line 2 onwards has the data. The columns are as follows:
    1. ***Country*** – the name of the country. The simple ebola data file has only one country, while the complex ebola data file has multiple countries.
    2. ***Locality*** – the name of the locality for which the data was collected. In the simple ebola data file, the locality is listed as “National” indicating that the data is for the whole country. In the complex ebola data file, there are a different set of localities for each country, including one called “National”, representing the entire country.
    3. ***Indicator*** – indicates what the value field represents. It is either `cumulative_cases` (measuring the number of cases from the beginning of the outbreak to date) or `cumulative_deaths` (measuring the number of deaths from the beginning of the outbreak to date)
    4. ***Date*** – the date on which the data was collected
    5. ***Value*** – a number indicating the cumulative number of either cases or deaths, as specified by the *Indicator* column

**Aside:** *Note that the data stored in the file is* ***cumulative*** *data, that is, the number of cases/deaths from the beginning of the outbreak to the indicated date. For task 1, questions (d) to (g) ask about the infection/death* ***rate****. For this purpose, we can define the infection rate at a particular date as the cumulative number of cases at that date minus the cumulative number of cases at the previous date when data was recorded, divided by the number of days between the two dates. For example, if there are 456 cases on 5th January and the previous number of cases recorded was 300 cases on 3rd January, the infection rate can be computed as (456-301)/(5-3) = 155/2=77.5.*

For task 2, the **partial time series** also has a .csv extension, but it has only one column of data, representing the cumulative number of cases of Ebola in an unspecified locality, for an unspecified period of time.

Note that the data files provided are simply samples. Your code will be tested on these and other data files with the same format, but which could have fewer or more data points than the sample input files that have been provided. As such, you need to write your code to be general enough to work with any number of rows of data.

## Output File Formats

For task 1, your programme should output two files named *task1_answers-<input_file>* and *task1_times-<input_file>* respectively, where *<input_file>* is the name of the input file. E.g. if your input file was ***data1.txt***, the output files will be named ***task1_answers-data1.txt*** and ***task1_times-data1.txt*** respectively.

- The file *task1_answers-<input_file>* should have answers to the questions (a) to (g) of task 1, one answer per line.
- The file *task1_times-<input_file>* should have the measured runtimes, in milliseconds, of different sections of your programme:
    - Line 1 should be the file-reading and pre-processing time
    - Lines 2-8 should be the time to answer questions (a) to (f) respectively
    - Line 9 should be the overall runtime of your programme.

For task 2, your programme should output one file: *task2_result-<input_file>*, *<input_file>* is the name of the file with the partial time-series data.

- The first line of file *task2_result-<input_file>* should have the name of the country and locality matching the partial time series data provided.
- The second line of the file should indicate whether the data corresponds to cases or deaths
- The third line of the file should indicate the starting date of the partial time series data provided.
- The fourth line of the file should have the measured runtime, in milliseconds, of your programme.

## What to submit

You should submit:

1. Your code
2. A readme.txt file indicating which tasks your code supports and any additional information needed to run your code.
3. A 1-page document briefly describing your high-level approach, and listing any references (see rules below).

## Rules of the Challenge

1. Participation in the competition is by Ashesi students, either as **individuals** or as a **team** made up of **two** students from the same yeargroup. **No collaboration** is allowed except between two

members of a team. Individuals who are not Ashesi students may not participate.
2. In addition to overall winners, there will be winners for each year group, so members of all year groups are encouraged to participate and not feel intimidated by the idea that they may be “competing” against more experienced programmers.
3. Solutions must be coded in Python or Java. Standard built-in classes and libraries can be used (e.g. the math library in Python and classes in the java.util package in Java). No external/third-party libraries can be used in either language.
4. All submitted code must be your own. Code may not be copied from any source.
5. You are allowed to do research if needed. However, you **must** cite **all** resources you consult.

## Evaluation Criteria

We reserve the right to disqualify submissions that do not follow the specified code structure or do not comply with the submission instructions.

Solutions will be evaluated according to the following criteria. The order below will be used to rank the various submissions.

1. Completion & correctness (the number of tasks solved correctly).
2. Efficiency (the time it takes to execute your solution approach). Accommodation will be made for the inherent difference in speed between Java and Python
3. Code formatting and structure: code should be modular, well-structured, well-formatted, and well-commented.
4. Clarity of 1-page documentation of approach