**Software Development Engineer in Test (SDET)**

**Candidate Brief**

**Overview of the Exercise**

In this exercise you are required to present your solution to the panel.

You will be required to deliver your findings to the panel orally and by Screen Share to deliver your presentation.

**Background Information**

A sample of records has been selected as part of a Proof Of Concept for a new service in the form of a CSV file.

This CSV contains data on driver records and needs to be parsed, validated, and stored in the correct format.

The CSV contains the following fields:

* driverID, firstName, lastName, dateOfBirth, entitlements

**Validation rules:**

*A field is deemed invalid unless it meets the criteria listed below.*

*All characters are case insensitive.*

**DriverID**

* Mandatory

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Last Name | | | | First Name | Month of Birth | | Year of Birth | |
| L | L | L | L | L | D | D | D | D |
| Letter | Letter | Letter | Letter | Letter | Digit | Digit | Digit | Digit |

* Last name
  + First four alphabetic characters of a person’s last name
  + If a person’s last name has less than four characters, then the character ‘X’ must be used to fill up the remaining space
  + Only alphabetic characters (A to Z) characters can be used to construct a DriverID. Special characters (hyphen or apostrophe) are skipped
* First name
  + Initial character of a person’s first name
  + If a person does not have a first name, then the character ‘X’ must be used as the initial
* Month of birth
  + Zero-padded
* Year of birth
  + Year without century

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **First Name** | **Last Name** | **Date of Birth** | **Driver ID** |
| Joe | Bloggs | 1/1/1990 | BLOGJ0190 |
| Jane | Doe | 25/3/1983 | DOEXJ0383 |
|  | Smith | 10/07/2001 | SMITX0701 |
| Joe | B’loggs | 1/1/1990 | BLOGJ0190 |

**First Name**

* Optional
* If present, must contain at least one alphabetic character
* Following special characters are allowed:
  + Hyphen
  + Apostrophe
* A person may have multiple first names

**Last Name**

* Mandatory
* Must contain at least one alphabetic character
* Following special characters are allowed:
  + Hyphen
  + Apostrophe
* A person may have multiple last names

**Date of Birth**

* Mandatory
* Day and Month can either be zero-padded or not
* Year must include century
* A person cannot be older than 100 years
* A person cannot be younger than 15 years
* Date must not be in the future

Examples:

* Valid: 1/1/1950, 01/01/1950, 1/01/1950
* Invalid: 1/1/50, 1/1/1900, 1/1/3000

**Entitlements**

* Mandatory
* One or more of the following:
  + A
  + B
  + C
  + D

Examples:

* Valid: [“A”], [“A”, “B”], [“C”]
* Invalid: [“X”], [], [“A”, “B”, “X”]

**Format:**

When the CSV data has been parsed and validated, it will need to be stored in the following format:

|  |  |
| --- | --- |
| **Driver ID** | LLLLLDDDD   * All capitals |
| **First Names** | * Title case |
| **Last Name** | * Title case |
| **Date of Birth** | 01, Mar, 2001   * Zero-padded day * Abbreviated month * Year with century * Comma separated |
| **Entitlements** | [Motorbike, Car, Lorry, Bus]   * A -> Motorbike * B -> Car * C -> Lorry * D -> Bus |

**Presentation Scenario**

You will need to author code which performs two separate functions:

1. Parse the CSV data file and determine how many valid and invalid records the file contains.
   1. For invalid records, the reasons why they are invalid should be present. e.g invalid fieldA
2. For the valid records, output their data in the correct format to either terminal or file.

You may use **any** reasonable general-purpose programming language to implement your solution. (Your audience will be unlikely to understand deliberately esoteric and obfuscatory toy languages, e.g., Malbolge.)

The code is expected to compile and run successfully. You may use third-party libraries to assist you, but the implementation of task must be original.

You **do not** have to implement testing for this code, **but** you should consider how you might test it since you may be asked about it.

**You must submit your solution at least 24 hours before your interview.** Please provide a link to a public source code repository such as GitHub to [ITSRecruitment@dvla.gov.uk](mailto:ITSRecruitment@dvla.gov.uk)

**The Presentation Task**

You should prepare a short (10 minutes) presentation in advance. You will be asked to present this during the interview.

Your slides and talk **should** cover:

* A working demo of your code against the two criteria
* A general overview of how your code works
* Aspects of the task you enjoyed or disliked
* Aspects of the task you found difficult and why
* Aspects of your code you would improve or change if you refactored it or did the task again

Your slides and talk **should not** cover:

* A line-by-line explanation of the code