SCANNED DOCUMENT INFORMATION PARSER

One of the most important services offered by Thales is the identification management. To perform that, Thales scans identification documents (such as passports, ID cards or driver licenses) and retrieves the personal information within them.

As a result of this process, we are trying to create an identification database, avoiding duplicate results (sometimes the same document is scanned twice) and filtering those documents that are not legit.

We are discarding those cases where the same person presents two different documents. We are considering that two documents refer to the same person if:

- The first name, last name, nationality and date of birth are the same

We are considering that two scans refer to the same document if:

- The document number, issuing country and document type are the same.

Remember, OCR is not an exact science and some typos can be included in the data extracted. Apart from that, some kind of documents are not accepted. Your code must be able to handle the following situations:

- Expired documents should be avoided. All dates in the document are represented by YYMMDD (i.e. April 10th 2005 will be represent as 050410). Wrong dates should result in discarded documents (you can assume that all dates are from this century).
- Issuing country and nationality are represented using ICAO three-letters format. If the field has more than 3 letters all letters after the first three should be ignored. We only accept Spanish (ESP) documents and those for neighbor countries. Spain is surrounded by France (FRA), Portugal (POR), Andorra (AND) and Morocco (MOR).

We need this detection code to run quickly.

The input could be large enough so that it will behoove you to make your code as fast as possible. That said, please remember that this identification system is part of a larger system and one that might change over time, and we expect the structure of your code to reflect that fact.

Programming challenge:

Design and code a solution for the problem described above. Please use **Java** as main language. It is important that you **apply your knowledge** on coding best practices, design patterns, clean code, test, etc. So try to polish it as much as you can.

The program will take lines, one by one, by reading console input. The attached file sample_input.txt contains an example of input. However, please remember that input is not file-based but console-based.

The solution will be pushed on GitHub on a public repository (use free plan, no need to pay for it). If necessary, you can include compilation instructions or deployment instructions, system requirements, usage, etc. We expect to see commit history when developing the solution on that repository, ensuring it does not contain any binary file.

The delivery of the challenge will be **the link to the GitHub repository**.

Input:

First line will contain an integer N denoting the number of records, followed by N lines with one record per line.

Each record contains the following information separated by commas:

- scan id (numeric)
- Document type (P|ID|DL)
- Issuing country
- Last name
- First name
- Document number (numeric)
- Nationality
- Date of birth
- Date of expiry

Output:

A comma-separated output with the discarded scan ids in ascending order and the discarding reason.

ANNEX:

- Sample input:

```
5
1,P,ESP,PEREZ,JAVIER,10011,ESP,020403,230510
2,DL,FRA,PEREZ,JAVIER,5454,ESP,020403,250510
3,P,ESP,GARCIA,JUAN,12011,HRV,021210,160510
4,ID,USA,GARCIA,MANUEL,12021,ESP,051210,190510
5,P,AND,GONZALEZ,LAURA,52301,ITA,161320,230511
```

- Sample output:

```
2,same person
3,expired
4,non-accepted country
5,invalid date
```

- Sample Explanation:

The second one should be discarded because refers to the same person, the third one should be discarded because the document is expired. The fourth one should be discarded because the document have been issued in a non-accepted country. The fifth one should be discarded because the date is not valid.