Universidad icesi	**Departamento de Computación y Sistemas Inteligentes** **Computación y Estructuras Discretas III** **Atomata theory and applications**

Objetivos

Unidad 1 – Lenguajes Regulares y Autómatas

Definir formalmente los distintos tipos de autómatas finitos (deterministas, no deterministas, no deterministas con transiciones lambda, transductores), dar ejemplos, identificar sus elementos y conocer sus aplicaciones.
Entender y aplicar, tanto las nociones de lenguajes y expresiones regulares, como de autómatas de estado finito, para el reconocimiento de patrones, procesamiento, validación y extracción de texto usando un lenguaje de programación.

The university has a dataset of students' personal information, including their date of birth. Some dates are badly formatted, and the administration wants to classify students by generation once the dates are validated. The dataset in the file `students.csv` contains the attributes id, name, dob (date of birth), and program. Note that, the file is a CSV with semicolon (;) as separator.

First challenge

Filter the dataset keeping only the students correctly formatted.

1. Design ((by hand and later digitalize) and implement in pyformlang a DFA to accept only dates in the ISO format: **`YYYY-MM-DD`.** Constraint: the date string must be exactly 10 characters long**.**
   Examples:
   `1999-04-25` → valid
   `2001/10/15` → invalid (wrong separator)
   `2008-9-5` → invalid (wrong length)
2. Use your automata to filter the dataframe to keep only rows where the DFA accepts the dob. Use a new dataframe called filtered_students.

Second Challenge

Classify students based on their **year of birth** (from the validated rows in the filtered_students dataframe). For this classification, use the following generational ranges:

- Baby Boomers → 1946–1964
- Generation X → 1965–1980
- Millennials → 1981–1996
- Generation Z → 1997–2012

1. Design (by hand and later digitalize) and implement in pyformlang the automaton corresponding to the generation assigned by your teacher. The automaton accepts strings where the first four characters (the year) fall into the corresponding range.
   Example:
   1960-02-01 → Baby Boomer DFA
   1985-07-30 → Millennial DFA
   2008-09-15 → Gen Z DFA

2. Use your DFA to filter the students in the assigned generation in a new dataset called using the generation's name and create a new column generation containing the category in the classification. To solve this, check the documentation and examples:
   Adding a new column to existing Dataframe in Pandas

Third Challenge (This is optional)

Identify students born on Leap Day (**February 29)**. This task should be performed over the original dataset, not the filtered ones produced on challenges one and two. To solve this challenge, consider the following:

- The original dataset contains multiple formats
  - YYYY-MM-DD → e.g., 2000-02-29
  - DD/MM/YYYY → e.g., 29/02/2004
  - Month DD, YYYY → e.g., Feb 29, 2008

1. Design (by hand and later digitalize) one automaton to accept each format. That is, you need to design one automaton per format in the spreadsheet.
2. Combine the designed automata (union) into a single automaton and present your design using a diagramming application.
3. Implement the resulting combined automaton using pyformlang.
4. Filter students whose dob matches Leap Day and keep then in a dataframe called LeapDay

Deliverables and points:

For each challenge, you must deliver the following artefacts:

1. [20 pts] The digitalized version of the transitions diagram for each designed automaton, this design must be created by hand and digitalized using a scanner app.
2. [20 pts] The code for the implementation of each automaton using Pyformlang, this implementation and the design must be consistent.
3. [5 pts] The code for the Dataframe filtering.
4. [5 pts] A set of tests for each solved challenge. Include at least 3 positive cases and 3 negative cases per automaton

The optional challenge has 50 points.