A file of the format contains zero or more **data elements** and optional **comments**. White space (referring here and in the following to either space characters or tabulations) are ignored, unless they are found within a string.

- each **data element** consists of a **key**, an **assignment sign** and a **value**, separated by zero or more white spaces.
  - **key** is an alphanumeric string (valid characters are uppercase and lowercase letters, digits, and the undescore symbol);
  - **assignment sign** is ">";
  - **value** can be of the following types:
    - *string*: any sequence of symbols (except double quotes), surrounded by a pair of double quotes;
    - *integer*: signed decimal (sign is present only for negative integers);
    - *boolean*: literal `TRUE` and `FALSE` (case sensitive, no quotes).

  Some examples of data elements:

  ```
  key1 > "value"
  key2>TRUE
  key3 > 1
  key4 > "1"
  ```

- a **comment** starts with a dollar sign ("`$`") (unless it is found within a string) and ends with the end of the line. Comments are ignored.

If a line is empty (or only contains comments), it is ignored. A line can contain one single data element, or more than one, separated by commas (","). A comma at the end of the line is valid and can be ignored.

Note If the format description gives rise to an ambiguity, make a choice to solve it and describe the choice you made in your solutions.

Exercise 2

Aim *Create a web page to graphically visualize points (whose coordinates are obtained from a REST API) inside a two-dimensional rectangle in a web page. Update the coordinates upon user request and make older points fade out.*

Description We provide a very simple API at the following address:

```
http://theossrv2.epfl.ch/aiida_assignment2/api/points/
```

A GET request to the above URL returns some data (in JSON format) that changes over time. The data you are going to obtain has the following format:

```
{
  "circles": [
    {
```

```
      "y": 246.61493308178768,
      "x": 59.55507262280179,
      "id": 0
    },
    {
      "y": 103.029296875,
      "x": 103.029296875,
      "id": 1
    },
    {
      "y": 90.20206845720121,
      "x": 156.40940475463867,
      "id": 2
    }
  ],
  "range": {
    "min_x": 0,
    "min_y": 0,
    "max_x": 300,
    "max_y": 300
  }
}
```

In particular, the `circles` list contains, for each point, the `x` and `y` coordinates of the point together with an integer `id` to uniquely identify it. The `range` dictionary contains the minimum and maximum value that the coordinates can assume (you can assume

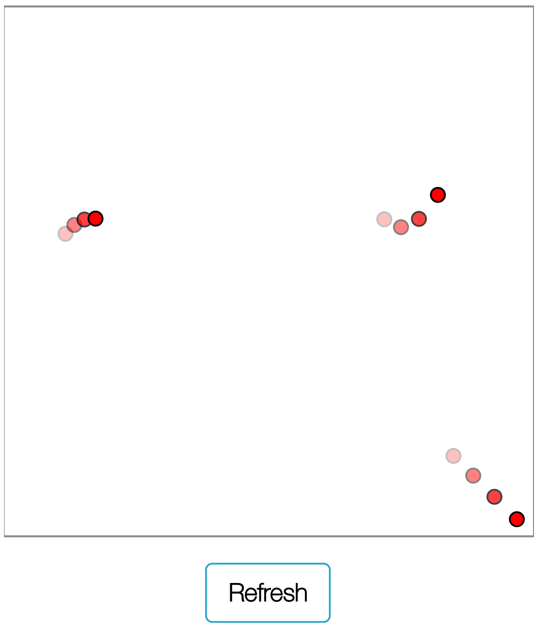

Figure 1 – Example screenshot of the webpage to be realized in task 2. The screenshot was taken after four successive clicks on the "refresh" button. If the button is not clicked, the webpage does not change.

that the values in the `range` dictionary are always integers in the range 150–1000, so that you can map `x` and `y` values directly to pixels).

> *You have to prepare an HTML web page that loads at launch time the coordinates of the points from the above API and visualizes each point as a red dot on the page.*

The page should also provide a "Refresh" button to reload the data dynamically from the API (without a full refresh of the page). When the button is clicked, the new data should be represented by red dots, but the old dots that were already present in the page must not disappear, but rather become more transparent. Successive pressures of the "Refresh" button will make new dots appear and have the old ones fade out (in the example shown in Fig. 1, the oldest points disappear completely after four successive clicks).

Note The appearance of the page is not required to be identical to the one shown in Fig. 1, if the functionality is the same as described above.

Note We did not put any filter on the number of connections per second to the API. We ask you to be careful not to overload the server (a rate of 10 connections per second is more than enough for the purposes of this exercise).

Exercise 3

Aim *Given the coordinates of a set of points on the plane, consider a link between two points if their distance is below a given threshold (obtaining in this way an undirected graph). Then, given a list of pairs of points in input, return whether each pair of points is connected or not through a path in the graph.*

Description Your code is going to receive in input a dictionary (in JSON format) with the following data: a `threshold` value, a list of `points` (each point is a dictionary containing an `x` and a `y` coordinate, as well as an integer `id`), and a list of pairs of points (each pair is a list of length two, where the two elements are the IDs of two points).

An input file therefore looks like this:

```
{
  "threshold": 120.0,
  "points": [
    {
      "y": 34.95169607701711,
      "x": 68.86475009715542,
      "id": 0
    },
    {
      "y": 252.6587436178828,
      "x": 297.62557538865025,
      "id": 1
    },
    {
```