Problem 1. Writing a polynomial to a file. Write a program that asks the user for i) the name of the file where to store our polynomial, ii) and the coefficients of a polynomial. The user must answer 'No' to stop introducing coefficients. Then the program writes to that file the coefficients, in a single line, using spaces as separators.

Problem 2. Reading and storing the values of `x`. Now, its time to save values of `x`. The program request to the user i) the name of the file where to save the values of `x`, ii) and all the values of `x` given by the user. The user must answer 'No' to stop introducing values of `x`. The program writes to that file the values of `x`.

Problem 3. Evaluating a polynomial. Now, the program must ask the user for the file where the coefficients have been stored and the file containing the values of `x`. The program must evaluate the polynomial for each value of `x`, printing both `x` and `p(x)` to the screen.

Problem 4. A second release of the polynomial case

Now we want to read several polynomial (once at a time), evaluate them at 0.5 and write to a file i) first the polynomial and then the evaluation, a polynomial in each line.

To read a polynomial coefficients, the program requests them to user one by one, starting with the coefficient at position 0 and ending whenever the user introduces 'STOP'. Store the coefficients in a list.

The expected writing of a polynomial to a file is defined as follows. Let's say one of the polynomial is $p(x) = 3 \times x^4 - 2.5 \times x^2 + x$; $p(0.5) = 0.0625$. Then, the corresponding line in the file must be:

```
p(x) = 3 * x ** 4 + 0 * x ** 3 - 2.5 * x ** 2 + x; p(0.5) = 0.0625
```

Reading polynomials from the user ends whenever the user introduces "HALT!".

Problem 5. The CIA is sneaking you! Reading a data file from the CIA official page (see this url: `https://www.cia.gov/library/publications/resources/the-world-factbook`. From that site we have downloaded two data sets: one concerning the countries and their population (file `rawdata_335.csv`) and a second one with the areas (file `rawdata_279.csv`); both files are Comma Separated Values (csv) files. In the former you can see several columns: the order, the country, the population and when the data was obtained; the rows are sorted in decreasing order of the population. In the second you can see the order, the country name, the surface and an empty field; all is sorted according to the area in decreasing order.

Now, you must write python program that allows you to read the data from the file and

1. Compute the total population (sum of all countries).

2. The average population per country (total inhabitants divided by number of countries).

3. Now, determine the country that is closest to this average population per country.

4. Create a file with the country, its population, its area and the population density.

Problem 6. This is a problem to be solved with writing and reading from file, strings and lists. We have a file with the results from each match during the half of the season. Here you have the results, copy and paste that to a file, for example, 'season_data.txt':

```
Wolves,2,Northants,1
Bubbles,1,Calla,2
Tomsom,2,Brown,0
Brown,0,Wolves,2
Northants,1,Bubbles,2
Calla,2,Tomsom,1
Wolves,2,Tomsom,1
Northants,1,Calla,2
Bubbles,1,Brown,2
Calla,2,Wolves,1
Northants,1,Brown,2
Tomsom,1,Bubbles,2
Wolves,2,Bubbles,0
Tomsom,1,Northants,2
Brown,2,Calla,1
```

Now, it's time to prepare some code!

1. From the 'season_data.txt' file we are going to obtain the list of team names and the list of scores for each match. This list of match's scores will be a list of list, a list per match; the list for a match includes the name of the local club (string), the number of sets this one won (integer), the foreigner club (string) and the number of sets won by this one (integer).

   Open the 'season_data.txt' file for reading and read the data. To do so, create an empty list called data and an empty list of names called clubs. For each line s, i) convert it to a list -using the method split, call this new list linedata-, ii) then convert to integer the values for the number of won sets, iii) if not contained in clubs, insert each club name in this club list, and iv) insert linedata in data. Remember to close the file.

   Now we have a list called clubs with the names of the clubs that participated in the season. And we have the list data with the results from each match. Examples are given for each list below... We will use this in the next exercise.

```
>>> clubs
['Wolves', 'Northants', 'Bubbles', 'Calla', 'Tomsom', 'Brown']
>>> data
[['Wolves', 2, 'Northants', 1], ['Bubbles', 1, 'Calla', 2],
['Tomsom', 2, 'Brown', 0], ['Brown', 0, 'Wolves', 2],
['Northants', 1, 'Bubbles', 2], ['Calla', 2, 'Tomsom',1],
['Wolves', 2, 'Tomsom', 1], ['Northants', 1, 'Calla', 2],
['Bubbles', 1, 'Brown', 2], ['Calla', 2, 'Wolves',1],
['Northants', 1, 'Brown',2], ['Tomsom', 1, 'Bubbles', 2],
['Wolves',2, 'Bubbles', 0], ['Tomsom', 1, 'Northants', 2],
['Brown', 2, 'Calla',1] ]
```

2. Remember that the list data contains the results for each match, and the list clubs contains the list of the teams in the league. Now we have to count the number of wins and losts for each club. As there are no possible draw, only the victories for each team are needed.

   Let's store this in the list wins, which has to be initialized to a list of zeros, as many as clubs played the league. To create this list, use `wins = [0] * len(clubs)`.

   This can be the algorithm: now we have to go match by match, determining who won and increasing in one the number of victories in the list wins. At the end, we have the number of wins for each club during the season.

```
# A L G O R I T H M   F O R   T H I S   T A S K
for each match_result r in data:
   club1 = r at position 0
   club2 = r at position 2
   score1 = r at position 1
   score2 = r at position 3
   if club1 won
      obtain the index i of this club within the list clubs
         MAKE USE OF METHOD index FOR LISTS!!!
         E.g.: mylist.index('Coco')
      increment the number of victories for position i in wins
   else :
      obtain the index i of club2 within the list clubs
      increment the number of victories for position i in wins
```

   At this point, the state of affairs is:

```
>>> clubs
['Wolves', 'Northants', 'Bubbles', 'Calla', 'Tomsom', 'Brown']
>>> wins
[4, 1, 2, 4, 1, 3]
```

3. We want to create a comma-separated-values file with the number of victories for each club. Create a file call 'mid_season_standing.csv', with each line containing the club name and the number of victories. The expected result is:

   ```
   Wolves,4
   Northants,1
   Bubbles,2
   Calla,4
   Tomsom,1
   Brown,3
   ```

Problem 7. Write a program that requests the name of a TXT file that contains a text in English. The program must be able to create an output file called `frequencies.txt` that contains the absolute frequency of each word in the text in the following way:

```
a;15213122
the;12312
other;543
```

To do that you must eliminate all symbols not present in the English alphabet and convert all words in lowercase.

Problem 8. Write a program that asks for several frequencies files and combines them up. The user must answer 'No' to stop introducing files. After combining the frequencies of all the files, the program must output the result to another file called `frequencies_all.txt` that follows the same format.

Problem 9. Write a program that asks for a frequencies file and outputs the same frequencies but using relative values instead of absolute ones to a file called `rel_frequencies.txt`. All the absolute frequencies in the output file must sum up to 1. The words in the output file must be sorted by frequency. The most frequent word must be the first one.

```
a;0.13
the;0.12
other;0.04
```