Rehearsal exercises

Exercise 1

In this program you use the file *books.txt.*

a) Write a function *read_books()* that returns a list with the books you find in the inputfile.

b) Write a function *menu* that returns the selection the user makes. When the user doesn't enter a, A, b, B, c, C, s or S, the function keeps showing the menu.

```
a - Overview
b - Longest title
c - 5 letters on a row
s - Stop
Make your choice:
```

c) Write a function *print_list(books)* that prints the list formatted like the screenshot below.

```
List of books:

1 Don Quichote
2 Things Fall Apart
3 In Search of Lost Time
4 Ulysses
5 The Great Gatsby
6 One Hundred Years of Solitude
7 Moby Dick
8 War and Peace
9 Hamlet
10 The Catcher in the Rye
```

d) Now write the main program. In this main program the menu is shown repeatedly until the user presses **s.** In that case, the program stops immediately.

- If the user selects **a in** the menu, the overview will be printed with the function you wrote in part c).

- If the user selects **b** in the menu, the overview is printed, followed by the length of the longest title.

```
List of books:

1 Don Quichote
2 Things Fall Apart
3 In Search of Lost Time
4 Ulysses
5 The Great Gatsby
6 One Hundred Years of Solitude
7 Moby Dick
8 War and Peace
9 Hamlet
10 The Catcher in the Rye

The longest title has 29 characters
```

- If the user chooses **c**, a book number is asked. Then the title of that book appears 5 characters at a time (with a space between every character).

```
    a - Overview
    b - Longest title
    c - 5 letters on a row
    s - Stop
    Make your choice: c
Enter booknumber max 10: 3

I n   S e
a r c h
o f   L o
s t   T i
m e
```

Exercise 2

This program uses *classlist.csv.* You write a tool to easily check who is present or absent in the lesson.

a) Write a function *control* with 1 parameter: the name of a class. In this function you read the file *classlist.csv* and show all the names of the students of that class.

The user indicates for each student of that class whether he/she is present or not. So enter N or n when the student is not there and otherwise just press Enter.

```
In which class do you want to do the check: 1 ACS 02
ADAM ABID:
MESUMBE AKOLLAH JUDE:
OGECHUKWU BALOGUN:
RUNE BREUGELMANS:
ANNELORE CALEPET: n
AMANDA COLLINS:
THIERRY EEMAN:
FREDERIK GIELIS:
LIAM HUYBERECHTS:
RUSTEM KAMALIDENOV:
EHRAN LENAERTS: N
SHARON MAHARJAN:
JELENA SANADER:
```

. The function returns a list of student names extended with

- OK: if the student is present
- NOT: if the student is not present

b) In the main program you first ask for which class you want to do the presence check. The function *control* is called and a list is returned.

You write this created list in the file with the same name as the class (for example "***1 ACS 02.csv***")

```
Attendance list 1 ACS 02
ABID;ADAM;OK
AKOLLAH JUDE;MESUMBE;OK
BALOGUN;OGECHUKWU;OK
BREUGELMANS;RUNE;OK
CALEPET;ANNELORE;NOT
COLLINS;AMANDA;OK
EEMAN;THIERRY;OK
GIELIS;FREDERIK;OK
HUYBERECHTS;LIAM;OK
KAMALIDENOV;RUSTEM;OK
LENAERTS;EHRAN;NOT
MAHARJAN;SHARON;OK
SANADER;JELENA;OK
```

If the you entered a class that doesn't exit, the list will be empty. In that case you don't create an output file but you show this error message:

```
In which class do you want to do the check: 1i 05
This class doesn't exit
```

Extra: write a function that returns a set with all possible class names. Use this set to help the user with the input of the class name.

You will read the input file twice:

- Look for all possible class names
- Look for the students in that class

Exercise 3

In a kindergarten they want to surprise a number of children with a sticker with a figure on it. They want to do this according to your programme, which will consist of two functions and a main programme.

In your programme you can use:

- the file *names.txt* in which you find the names of 5 children on each row, separated by a /
- the file *figures.xml* that lists the shape and colour of all available figures.

a) Write a function ***read_names()*** that fills a list with the data from the textfile *names.txt.* For each row, one of the names is chosen randomly and that name is placed in the list.

At the end you sort the list alphabetically and return it.

```
['Anton', 'Bram', 'Ferre', 'Jarne', 'Jelle', 'Jens', 'Joe', 'Jordy', 'Kjelle', 'Klaus',
'Michael', 'Senne', 'Thomas', 'Tobias', 'Vincent', 'Wouter', 'Yannick'].
```

b) Write a function ***read_figures****()* that *reads* the xml-file *figures.xml* and returns a list to the main program. The list you need to create looks like this:

```
['red triangle', 'red square', 'red circle', 'yellow triangle', 'yellow square',
'yellow circle', 'blue triangle', 'blue square', 'blue circle', 'green triangle',
'green square', 'green circle', 'white triangle', 'white square', 'white circle']
```

c) Now write the main program using the two functions above to divide the figures.
Loop through all the names and then determine which figure that person gets. To determine which figure, use the length of the name.

For example:
If a person has a name consisting of 7 letters, he/she will get the 7th figure (blue triangle).
Of course, this also means that everyone with a name consisting of, for example, 4 letters will get the same figure (i.e. yellow triangle, the 4th figure from the list).

You print the overview in the main programme. For example:

```
A figure has been chosen for the following toddlers:
Arno     yellow triangle
Ben      red circle
Florian      blue triangle
George   yellow circle
Guylian      blue triangle
Jef      red circle
Jonathan     blue square
Jules    yellow square
Klaus    yellow square
Kobe     yellow triangle
Mats     yellow triangle
Mickey   yellow circle
Nicholas     blue square
Reinhard     blue square
Thomas   yellow circle
Vince    yellow square
Willem   yellow circle
```

Exercise 4

In this exercise you use the file *prices.txt* in which you can find prices of a number of basic products in various shops.

a) Write a function ***lowest_price()*** that reads the data from the file and returns a dictionary as return value to the main program. This dictionary contains the name and the lowest price for each product in the file. Use the technique of data grouping.

b) Now write the main program using the function above to print this list.

```
Price list
----------
milk      0.97
water     0.39
coffee    1.55
tea       2.39
speculaas     2.85
cornflakes    3.15
butter    1.55
```

c) Complete your main programme so that someone can easily find out how much the lowest price is for one or more products. To stop reading product names, enter an x or X.

For example:

```
Enter the item (press x if you want to stop): milk
The lowest price of milk is 0.97 EUR
What do you want to know the price of (press x if you want to stop): tea
The lowest price of tea is 2.39 EUR
What do you want to know the price of (press x if you want to stop): x
```

If the user entered an incorrect name, the question is asked again until the user enters the correct name.

```
Enter the item (press x if you want to stop): koffee
This item is not available.
Enter the item (press x if you want to stop): coffe
This item is not available.
Enter the item (press x if you want to stop): coffee
The lowest price of coffee is 1.55 EUR
Enter the item (press x if you want to stop): X
```

Exercise 5

Currently, we have a Discord bot running that collects user data from players playing the game of Overwatch. This data is stored in the files 'users.xml' and 'games.csv'. You can download them from Canvas.

The file 'users.xml' contains information about users that play the game. For each user the element 'userID' contains a unique identifier with which other files can refer to that user. The element 'battleTag' contains the name of the user.

The file 'games.csv' contains four columns with data about games played by certain users. The first column (with name 'UserID') contains the unique identifier of the user who played a game. The values in this field match with the field 'userID' of the file 'users.xml'. The second column (with name 'Tank') contains the score obtained by the user in the game for the role 'Tank'. The same principle applies for the columns 'Damage' and 'Support'. The values in these columns represent the score for the roles 'Damage' and 'Support' respectively. The games are ordered chronologically over time and listed correctly.

Step 1: read files (functions)

Create a function 'read_txt' with one parameter called 'file_name' in which you read the content of the file of which you pass the file name as parameter when using the function in your main program. The return value of this function is an array containing the data from the file you read in the function.

Create another function called 'read_xml' with the parameter 'file_name'. This function should read the XML file passed by the parameter of the function. The return value of this function should be a nested list which contains the user ID's and user names of the XML file. In your main program you may print the list to check whether its structure is good or not. If you choose to do so, the structure should look like this:

```
[['1', 'MystWizard'], ['2', 'brammetje'], ['3', 'Toontinus'], ['4', 'PVC42069'], ['5', 'Horcion'], ['6', 'Ardipithecus'], ['7', 'TheEconomist'], ['8', 'Carbxn']]
```

Step 2: visualize results (main program)

Your main program should start by calling the function read_txt and capturing its return value in an array called 'games'. Next, call the function read_xml and save its return value in a list called 'users'.

Create a visual representation of game results for each user in the list 'users'. Note that games are already sorted based on the field 'UserID' (in the same order as users are sorted in the XML file 'users.xml').

For each user, create a visualization with width 10 and height 3, divided in 3 separate line charts.

The first line chart should show the evolution of the scores for the role 'Tank'. Make sure the line of this line chart is shown in red and show a label with the caption 'Tank'.

The second line chart should show the evolution of the scores for the role 'Damage'. Make sure the line of this line chart is shown in blue and show a label with the caption 'Damage'.

The third line chart should show the evolution of the scores for the role ‘Support’. Make sure the line of this line chart is shown in green and show a label with the caption ‘Support’.

Show the user name for which data is shown as the title for the figure. **Make sure the title is shown in bold.** (search for the correct parameter online!)

Below you can find two examples (for the users ‘brammetje’ and ‘Ardipithecus’ respectively) of how your visualizations should look like.