14BHD COMPUTER SCIENCES, AA 2020/2021

Laboratory exercise 13

<u>Goals of the exercise</u>

- Practice several concepts of Python language to develop programs of intermediate complexity
- Explore different solutions to a proposed exercise.
- Use of complex structures for data storage

<u>Technical contents</u>

- Creation and use of:
    - Parametric functions
    - Variables and List
    - Sets and Dictionaries

To be solved in the laboratory

Exercise 1. A bowling match consists of ten frames. Each frame except for the tenth consists of one or two balls, or attempts to knock down the ten pins at the end of the alley. Doing so on the first ball of the frame is called a strike, and the second ball of the frame is not rolled. Knocking down all ten pins with both balls (having left some up with the first ball) is called a spare. If both attempts to knock down the pins leave some standing, the frame is called an open frame. A spare in the tenth frame gives the bowler one extra ball; a strike in the tenth gives him or her two extra balls.

A bowling score is computed as follows. A strike counts as 10 points plus the sum of the next two balls. A spare counts as 10 points plus the next ball. Any other balls merely count as themselves, as do any bonus balls rolled as a result of a strike or a spare in the tenth frame. The sequence of balls is stored in a text file, named *input.txt.* Suppose for example that the file contains the following sequence of balls:

```
9 1   0 10   10   10   6 2   7 3   8 2   10   9 0   9 1   10
```

You have to print the summary on screen. For instance, the score for the ten frames would be:

```
Frame   score
-----   -----
 1       10
 2       30
 3       56
 4       74
 5       82
 6      100
 7      120
 8      139
 9      148
 10     168
```

Write a Python program to read from file the scores for a sequence of balls and output the scores for the ten frames. There may be multiple lines of input, where each input line will be the scores for one player. The scores will be separated by one or more blanks. You may assume that the number of scores on the line is valid.

Exercise 2. Propose three different functions (programs) that find all numbers which are divisible by 7 but are not a multiple of 5, between 2000 and 3200 (both included). For each proposed function, the numbers obtained should be stored in a text file printed in a comma-separated sequence on a single line. Create a Python function that compares the three generated files and prints if all were equal or not.

Exercise 3. Write a Python program that can store a polynomial such as

$$\mathbf{p}(x) = 12x^{10} + 9x^{7} - x - 10$$

as a list of terms. A term contains the coefficient and the power of x. For example, you would store p(x) as

$$(12,10),(9,7),(-1,1),(-10,0)$$

Supply functions to add, multiply, and print polynomials. Supply a function that makes a polynomial from a single term. For example, the polynomial p can be constructed as

```
p = newPolynomial(-10, 0)
addTerm(p, -1, 1)
addTerm(p, 9, 7)
addTerm(p, 12, 10)
```

Then compute p(x) × p(x).

```
q = multiply(p, p)
printPolynomial(q)
```

Provide a module for the polynomial functions and import it into the driver module.

Exercise 4. Generalize the functions of *Exercise 3* and generate a function **Polycalc()** to perform the operation of several polynomials. Please, select a suitable number of input and output parameters for the function.

The input polynomials are stored in an input text file with the format:

Coefficient > power of X

5 > 10, 9 > 7, –1 > 1, –10 > 0

Check the operation of the function with the following example files:

13.5 > 126, 89.34 > 13, -12.23 > 4, –120.3 > 1
0.002 > 1, -0.23 > 0.85, 0.085 > 0.5, –0.068 > 0.25
0.4 > 8, 0.99 > 3, –1.35 > 12, –1000.03 > 0.5
12.4 > 0, 9.0 > 5, –1.2 > 12, –10 > 0
–23.90 > 6, –9.29 > 12, –12 > 4, –10 > 20, 1.3 > 1, 0.19 > 5, 1.95 > 7, 10 > 0