Computational Thinking and Programming – A.Y. 2019/2020

Written examination – 09/07/2020

Given name:	
Family name:	
Matriculation number:	
University e-mail:	
Group name:	
Is it your first try?	Yes | No

The examination is organised in three different sections:

- Section 1: basic questions [max. score: 8]. It contains four simple questions about the topics of the whole course. Each question requires a short answer. Each question answered correctly will give you either 2 points (full answer) or 1 point (partial answer).
- Section 2: understanding [max. score 4]. It contains an algorithm in Python, and you have to report the particular results of some of its executions according to specific input values.
- Section 3: development [max. score 4] It describes a particular computational problem to solve, and you are asked to write an algorithm in Python for addressing it.

You have 1 hour and 30 minutes for completing the examination. By the final deadline, you should deliver only the original text (i.e. this document) with the definitive answers to the various exercises that must to be written with a pen – pencils are not permitted. You can keep all the draft papers that you may use during the examination for your convenience – blank sheets will be provided to you on request.

Section 1: basic questions

1 – Which of the following are the steps that characterise the divide and conquer algorithms:

- base case, recursive step
- base case, divide, conquer, combine
- last in, first out
- base case: solution exists, base case: address directly, divide, conquer, combine, memorize
- leaf-win, leaf-lose, recursive-step

2 – Consider the following snippet of Python code:

```
def ln(inp, val):
    for p, i in enumerate(inp):
        if i != val:
            return p
```

Which value is returned calling the function above as follows: `ln(["a", "b", "c"], "b")`?

3 – Write down a small function in Python that takes in input two booleans and returns `True` if both are false, while it returns `False` otherwise.

4 – What are the main methods and the constructor to work with graphs using the NetworkX package available in Python?

Section 2: understanding

Consider the following functions written in Python:

```
def mir(family_name, mat_number):
    r = 0
    e = True
    for n in mat_number:
        if e:
            r = r + int(n)
            e = False
        else:
            r = r - int(n)
            e = True

    if r < 0:
        r = r * -1

    if r > 0 and family_name != "":
        idx = len(family_name) % r
        c = family_name[idx]
        return c + mir(family_name[1:-1], mat_number[1:-1]) + c
    else:
        return ""
```

Consider the variables `my_family_name` containing the string of your family name in **lower case**, and the variable `my_mat_number` containing the **string** of your matriculation number (ten digits). What is the value returned by calling the function `mir` as shown as follows:

```
mir(my_family_name, my_mat_number)
```

Section 3: development

Bibliograms are verbal constructs made when noun phrases included in a text are ranked high to low by their frequency of co-occurrence with one or more user-supplied seed terms. Each bibliogram has three components:

- a seed term that sets a context;
- tokens (i.e. strings separated by a space) that co-occur (i.e. precede and follow) with the seed across the text;
- counts (frequencies) by which co-occurring tokens can be ordered high to low.

For instance, let us consider the following text:

> *After a while, finding that nothing more happened, she decided on going into the garden at once; but, alas for poor Alice! when she got to the door, she found she had forgotten the little golden key, and when she went back to the table for it, she found she could not possibly reach it: she could see it quite plainly through the glass, and she tried her best to climb up one of the legs of the table, but it was too slippery; and when she had tired herself out with trying, the poor little thing sat down and cried.*

The bibliogram of the seed term "she" considering the text above is represented by the following list of tuples, each defining the token that co-occur with "she" accompanied by the related co-occurrence frequency: `[("found", 4), ("when", 3), ("had", 2), ("could", 2), ("went", 1), ("happened,", 1), ("decided", 1), ("door,", 1), ("it,", 1), ("it:", 1), ("and", 1), ("tried", 1), ("got", 1)]`.

Write an algorithm in Python – `def bibliogram(text, seed)` – which returns the bibliogram of the seed given the input text.