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

Written examination – 22/06/2020

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 ones are *not* an algorithmic technique:

- Backtracking
- Brute force
- Fortran
- Graph
- Greedy

2 – Consider the following snippet of Python code:

```
def xor(b1, b2):
    if b1 or b2:
        return not b1 or not b2
    else:
        return False
```

Which value is returned by calling the function above as follows: `xor(True, True)`?

3 – Write down a small function in Python that takes in input a string and a boolean and return a list of the *vowel* characters (i.e. those matching with any of the following ones: `"a"`, `"e"`, `"i"`, `"o"`, `"u"`) in the input string if the input boolean is `True`, otherwise (i.e. the input boolean is `False`) it returns a list of the characters that are *not* vowels.

4 – What are the main differences between *divide and conquer* and *dynamic programming*?

Section 2: understanding

Consider the following functions written in Python:

```
def main(f_name, mat_number):
    f_name_l = list()
    for c in f_name:
        f_name_l.append(c)

    mat_number_l = list()
    for idx, n in enumerate(mat_number):
        mat_number_l.append((idx, int(n)))

    prep(f_name_l, mat_number_l)
    return r(f_name_l)


def prep(name_l, mat_l):
    l_len = len(name_l)
    for p1, p2 in mat_l:
        if p1 < l_len and p2 < l_len:
            tmp = name_l[p1]
            name_l[p1] = name_l[p2]
            name_l[p2] = tmp


def r(ipt):
    result = 0
    cur_len = len(ipt)
    if cur_len == 0:
        return result
    else:
        el = ipt[0]
        if el in "aeiou":
            result = cur_len
        return result + r(ipt[1:])
```

Consider the variable `my_family_name` containing the string of your family name in **lower case** and **without spaces**, and the variable `my_mat_number` containing the string of your matriculation number (ten digits). What is the value returned by calling the function `main` as shown as follows:

```
main(my_family_name, my_mat_number)
```

Section 3: development

Co-citation is a similarity measure that uses citation analysis to establish a similarity relationship between documents. Co-citation is defined as the *frequency* with which two documents are cited **together** by other documents. The *co-citation index* of two given documents is higher depending on how many documents cite them both. For instance, suppose that document A cites documents D, E, F, G, document B cites documents F, H, I, and document C cites documents D, F. Thus, documents D and F have a co-citation index of 2, since both are cited together by documents A and C.

Write an algorithm in Python – `def co_citation(doc_1, doc_2, list_of_docs)` – which returns the co-citation index of the input documents `doc_1` and `doc_2` calculated from documents in the input list, weighted on the total number of documents available in the input list. Each item in the input list, representing a document, is defined as the set of the documents (identified by strings) it cites. For instance, if we call the function with input documents `"A"` and `"C"` and list of documents `[{"A", "B", "C"}, {"B", "D"}, {"A", "C", "E"}]`, it will return `0.66` as result, since the input documents are cited together in the 1st and 3rd documents in the input list and the overall number of documents in the input list is `3` – thus the weighted co-citation index is `2/3`, which is equal to `0.66`.