Computational Thinking and Programming – A.Y. 2020/2021

Written examination – 29/01/2021

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 algorithmic techniques are based on both the steps *basic case* and *combine*?

- divide and conquer
- greedy
- backtracking
- dynamic programming
- brute-force

2 – Consider the following snippet of Python code:

```
def f(s1, s2, n):
    if s1 < s2:
        return n
    else:
        return f(s2, s1, n * -1)
```

Which value is returned by calling the function above as follows: `f("mickey","donald",7)`?

3 – Write down a small function in Python that takes in input two numbers and a boolean value and returns the multiplication of the two numbers if the boolean is *True*, otherwise it returns the integer division of the two numbers.

4 – Describe which are the main components that define a Turing Machine.

Section 2: understanding

Consider the following functions written in Python:

```
def f(mat, name):
    name_l = list(name)

    uni = set()
    for i in mat:
        if i < len(name):
            uni.add(i)

    if len(uni) % 2 > 0:
        uni.remove(0)

    sl = list()
    for i in uni:
        pos = 0
        for j in sl:
            if j < i:
                pos = pos + 1
        sl.insert(pos, i)

    sl_last = len(sl) - 1
    for i in range(len(sl) // 2):
        s = sl[i]
        e = sl[sl_last]
        tmp = name_l[s]
        name_l[s] = name_l[e]
        name_l[e] = tmp
        sl_last = sl_last - 1

    return "".join(name_l)
```

Consider the variable `my_mat_l` containing a list of integers where each number is a digit of your matriculation number (e.g. `[0, 0, 0, 0, 1, 2, 3, 4, 5, 6]`), and the variable `my_fn` containing the string of your full name (i.e. given name and family name separated by a space) but in lower case. What is the value returned by calling the function `f` as shown as follows:

```
f(my_mat_l, my_fn)
```

Section 3: development

The **simple matching coefficient (SMC)** is a statistic used for comparing the similarity and diversity of two objects. In particular, given two objects *A* and *B*, represented by two dictionaries with *n* key-value pairs where the values can be set to either *True* or *False*, SMC is defined as:

$$\frac{N_{TT}+N_{FF}}{N_{TT}+N_{FF}+N_{TF}+N_{FT}}$$

where:

- N_{TT} is the total number of identical keys having value set to *True* in both A and B;
- N_{FF} is the total number of identical keys having value set to *False* in both A and B;
- N_{TF} is the total number of identical keys having value set to *True* in A and *False* in B;
- N_{FT} is the total number of identical keys having value set to *False* in A and *True* in B.

Write an algorithm in Python – `def smc(a, b)` – which takes in input two dictionaries having the same number of key-value pairs and all the keys in common, and returns the related SMC.