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

Written examination – 30/01/2020

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 – Select the item(s) of the following list referring to a *full digital* computer:

- Analytical Engine
- ENIAC
- Enigma
- Difference Engine
- Bombe
- Turing machine

2 – Consider the following snippet of Python code:

```
def f(s, n):
    if n < 0:
        return s
    else:
        return s + f(s, n-1)
```

Which value is returned by calling the function above as follows: `f("42", 1)`?

3 – Write down a small function in Python that takes in input two strings and returns the set of all the digit characters they do not have in common.

4 – Write down and explain what are two main characteristics that a computational problem should show to be sure that the application of a greedy approach will bring to an optimal solution to the problem.

Section 2: understanding

Consider the following functions written in Python:

```
def run(fn, mat_l):
    tot = 0 - len(fn)
    for n in mat_l:
        tot = tot + n

    return work(fn[0:tot], tot)


def work(s, n):
    if n > len(s):
        n = len(s)

    l = list()
    for c in s:
        if c not in l:
            l.append(c)
        else:
            n = n - 1

    idx = len(l) - n

    if idx < len(l) / 2:
        cur1 = l[idx]
        cur2 = l[n - 1]

        if cur1 > cur2:
            l.remove(cur1)
            l.remove(cur2)
            l.insert(idx, cur2)
            l.insert(n - 1, cur1)

        return work("".join(l), n - 1)
    else:
        return s
```

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 family name but in lower case. What is the value returned by calling the function `run` as shown as follows:

```
run(my_fn, my_mat_l)
```

Section 3: development

In cryptography, a **Caesar cipher** is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, C would become Z, D would be replaced by A, E would become B, and so on.

Write an algorithm in Python – `def caesar_cipher(msg, left_shift, shift_quantity)` – which implements the *Caesar cipher*, where `msg` is a lowercase string representing the message to encrypt, `left_shift` is a boolean that say if the shift should be performed on the left (*True*) or on the right (*False*), and `shift_quantity` is a positive integer that indicates how many positions one has to shift. The function must return the string of the encrypted version of the input message. Only plain alphabetic characters must be encrypted, all the others (punctuation, spaces, etc.) stay as they are.