Computational Thinking and Programming – A.Y. 2021/2022

Written examination – 15/07/2022

Given name:	____________________
Family name:	____________________
Matriculation number:	____________________
University e-mail:	____________________
Enrolment a. year:	[ ] 2021/2022 [ ] 2020/2021 [ ] 2019/2020 [ ] 2018/2019 [ ] other
Is it your first try?	Yes | No

The examination is organised in three different sections:

- Section 1: basic questions [max. score: 16]. 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 4 points (or less for partial answers).
- Section 2: understanding [max. score 8]. 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 8] 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 items in the following lists are **not** high-level programming languages?

- Python
- Assembly
- Machine language
- Cobol
- Fortran

2 – Consider the following Python function:

```
def f(x, y):
    if x <= 0 and y != 0:
        return x / y
    else:
        return y / x
```

What is the value returned by `f(3, 0)`?

3 – Write down a small function in Python that takes in input a string and a non-negative integer and returns `True` if the character in the input string at the position indicated by the integer is a vowel, otherwise it returns `False`.

4 – Introduce what are the main differences between the algorithmic approaches *brute-force* and *divide and conquer*.

Section 2: understanding

Consider the following functions written in Python:

```
def chk(name, mat):
    result = list()

    l_mat = len(mat)
    if l_mat > 0:
        max = l_mat // 2

        for idx in range(max):
            c = name[idx % len(name)]
            result.append(c)

        result.extend(chk(name, mat[:max]))

    return result
```

Consider the variable `my_mat` containing the list of all the ten numbers in your matriculation number (e.g. `[0, 0, 0, 0, 1, 2, 3, 4, 5, 6]`), and the variable `my_name` containing string of your given name all in lowercase. What is the value returned by calling the function `chk` as shown as follows:

```
chk(my_name, my_mat)
```

Section 3: development

Naive string search is a simple and inefficient way to see whether one string occurs inside another string. It proceeds to check the character at each index, one by one. First, we see if there is a copy of the string *S* starting at the first character of the string *T*. If not, we see if there is a copy of *S* starting at the second character of *T*. And so forth. For instance:

```
S: "aaa"             S: "baa"             S: "bbb"
T: "aaaaa"           T: "aababaa"         T: "aaaaa"
S in T: "aaaaa"      S in T: "aababaa"    S in T: "aaaaa"
```

Write an algorithm in Python – `def naive_ss(s, t)` – which takes two strings `s` and `t` and returns a tuple of two non-negative integers indicating the start and end position of `s` in `t` if `s` is contained in `t`, otherwise it returns `None`. For instance:

`naive_ss("aaa", "aaaaa")` will return `(0, 2)`

`naive_ss("baa", "aababaa")` will return `(4, 6)`

`naive_ss("bbb", "aaaaa")` will return `None`