**Computational Thinking and Programming – A.Y. 2018/2019**

Written examination – 29/10/2019

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 2 points.
- Section 2: understanding [max. score 4]. It contains an algorithm in Python, and you have explain what it does and 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 algorithms is based on recursion:

- Merge sort
- Insertion sort
- Linear search
- Quick sort
- Line wrap

2 – Please consider the following table of instructions of a Turing Machine, and the related tape initialised with all *0*s.

| Current state | Tape symbol | Write symbol | Move head | Next state |
|---|---|---|---|---|
| A | 0 | 1 | right | B |
| A | 1 | 0 | right | B |
| B | 0 | 1 | left | A |
| B | 1 | - | - | - |

Writing down the values written in the first three cells of the tape, starting from the one where the head was initially positioned.

3 – Write down a small function in Python that takes in input two strings and returns a set of all the characters they have in common.

4 – Explain when it is possible to use a recursive approach to solve a computational problem, and explain what is the main structure of a recursive algorithm.

**Section 2: understanding**

Consider the following function written in Python:

```
def test(gn, fn, mn):
    result = 0

    c_gn = cnt(gn)
    c_fn = cnt(fn)

    for c in c_gn:
        if c in c_fn:
            result = result + (c_gn[c] - c_fn[c])

    idx = (len(gn) + len(fn)) % len(mn)
    return result * (int(mn[idx]) + 1)


def cnt(s):
    result = {}

    for c in s:
        if c not in result:
            result[c] = 0
        result[c] = result[c] + 1

    return result
```

Consider the variable `my_gn` containing the **string** of your given name as written in the first page but in **lowercase** and **without** any space, the variable `my_fn` containing the **string** of your family name as written in the first page but in **lowercase** and **without** any space, and the variable `my_mn` the **string** of your matriculation number as written in the first page. What is the value returned by calling the function `test` as shown as follows:

```
test(my_gn, my_fn, my_mn)
```

**Section 3: development**

The function *all tokens string generation* is the process of creating the set of all the possible strings one can create combining, in a strict **alphabetical** order, either some of or all the tokens (i.e. strings) in a particular input seed (possibly unordered) list. For instance, considering the list of tokens containing the strings "home", "ball", and "sea", the set of all the possible strings of tokens one can generate includes the strings "ball", "home", "sea", "ball home", "home sea", and "ball home sea". Please note that the string "ball see" is not included in the set to return because, even if the tokens in such string are ordered alphabetically, they are not strictly ordered since there is another token, i.e. "home", which would have followed the token "ball".

Write a function in Python – `all_tokens_string_gen(tokens)` – that takes in input the list of tokens (`tokens`), and that returns the set of all the strings that one can generate using all the tokens as illustrated above.