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

Written examination – 07/07/2021

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 sentences are true statements?

- A queue is a countable sequence of ordered and repeatable elements.
- A set is a countable sequence of ordered and unrepeatable elements.
- A dictionary is a countable collection of unordered key-value pairs, where the key is non-repeatable in the dictionary.
- A list is a countable sequence of ordered and unrepeatable elements.
- A stack is a countable collection of unordered key-value pairs, where the key is non-repeatable in the dictionary.

2 – Identify all the mistakes introduced in the following implementation in Python of the Fibonacci platform, and propose the corrections to them.

```
def fib_dc(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 0
    else:
        return fib_dc(n-1) + fib_dc(n-3)
```

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

4 – Describe what are the steps characterising the *backtracking* algorithmic technique.

Section 2: understanding

Consider the following functions written in Python:

```
def r(f_name, m_list):
    f = list()

    for item in m_list:
        l_name = len(f_name)
        if item and l_name:
            idx = item % l_name
            f.extend(r(f_name[0:idx], m_list))
            f.insert(0, f_name[idx])
            return f

    return f
```

Consider the variable `my_mat_list` containing the list of the numbers in your matriculation number (e.g. `[0, 0, 0, 0, 1, 2, 3, 4, 5, 6]`), and the variable `my_full_name` containing string of your full name (i.e. given name plus family name separated by a space), all in lowercase. What is the value returned by calling the function `r` as shown as follows:

```
r(my_full_name, my_mat_list)
```

Section 3: development

The **Levenshtein distance** is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

The Levenshtein distance *lev(a, b)* between two strings *a* and *b* is 0 if one or both the strings are empty, and it is equal to *lev(a[1:], b[1:])* if the first characters of both the strings are the same. Otherwise, it is equal to 1 plus the minimum Levenshtein distance between these three cases:

- *lev(a[1:], b)*
- *lev(a, b[1:])*
- *lev(a[1:], b[1:])*

Write a recursive algorithm in Python – `def lev(a, b)` – which takes in input two strings `a` and `b`, and returns the Levenshtein distance.