Computational Thinking and Programming – A.Y. 2023/2024

Written examination – 11/01/2024

Given name: _____

Family name: _____

Matriculation number: _____

University e-mail: _____

Enrolment a. year: [ ] 2023/2024 [ ] 2022/2023 [ ] 2021/2022 [ ] 2020/2021 [ ] 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 – Being the variable *s* a set, which of the following Python codes are valid instructions?

- `s.append(127)`
- `s.add("hello")`
- `s.delete(42)`
- `s[2]`
- `range(len(s))`
- `s["a"]`

2 – Consider the following Python function:

```
def f(a, b, c):
    r = 0
    if a < b:
        for i in c:
            r = r + 1
    return r
```

What is the value returned by `f("alice", "bob", "charlie")`?

3 – Write down a small function in Python that takes in input a string and returns the number of characters that are vowels.

4 – Describe how it works the dynamic programming approach and introduce the application of its steps describing an exemplar algorithm (e.g. Fibonacci).

Section 2: understanding

Consider the following functions written in Python:

```
def ann(given_name, family_name):
    if len(given_name) + len(family_name) > 0:
        d = {}
        for c in given_name:
            if c not in d:
                d[c] = 1
            else:
                d[c] = d[c] + 1

        for c in family_name:
            if c in d:
                d[c] = d[c] - 1

        idx = -1000
        for c in d:
            if d[c] > idx:
                idx = d[c]

        p_char = set()
        for c in d:
            if d[c] == idx:
                p_char.add(c)

        n_given_name = []
        for c in given_name:
            if c not in p_char:
                n_given_name.append(c)

        n_family_name = []
        for c in family_name:
            if c not in p_char:
                n_family_name.append(c)

        return idx + ann("".join(n_given_name), "".join(n_family_name))
    else:
        return 0
```

Consider the variables `my_given_name` and `my_family_name` containing, respectively, the strings of your given name and family name in lowercase. What is the value returned by calling the function `ann` as shown as follows:

```
ann(my_given_name, my_family_name)
```

Section 3: development

A **rolling hash** function is a hash function able to convert an input string into an integer representing that string. It is based on multiplications and additions that depend on the number of characters in the input string. In particular, given a string *s* containing *n* characters, the hash value is computed by summing the multiplication involving the integer representation of each character and a coefficient *a* raised to the power of *n* minus the position number next to the current character, i.e.:

$$c_0 \cdot a^{n-1} + c_1 \cdot a^{n-2} + c_2 \cdot a^{n-3} + ... + c_{n-1} \cdot a^{n-n}$$

where *n* is the length of the input string *s*, *a* is a given constant value, c_0 is the integer representation of the character in position 0 of the input string *s*, c_1 is the integer representation of the character in position 1 of the input string *s*, c_2 is the integer representation of the character in position 2 of the input string *s*, etc.

Write an algorithm in Python – `def rolling_hash(s, a)` – which takes in input a string `s` representing a text and an integer value `a` representing a coefficient used in the formula above, and returns the hash value of that string computed as shown above. Python makes available the builtin function `ord` that takes in input a string of one character and returns the integer representation of that character – e.g. executing `ord("a")` will return the value 97.