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

Written examination – 17/10/2022

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 the items in the following lists refer to Python constructs?

- `return`
- `unsigned`
- `catch`
- `for`
- `foreach`
- `until`

2 – Consider the following Python function:

```
def f(x):
    r = 0
    x_len = len(x)
    while x_len > 0:
        r = r + x_len
        x_len = x_len - 1
    return r
```

What is the value returned by `f("me")`?

3 – Write down a small function in Python that takes in input two integers and returns `0` if they are equal, `-1` if the first is divisible by the second (i.e. the rest of the division is `0`), `1` if the second is divisible by the first (i.e. the rest of the division is `0`), otherwise it returns `None`.

4 – Introduce the *backtracking* approach and explain what is the data structure that usually is adopted for tracking the moves in the *peg solitaire* problem introduced in the text book.

Section 2: understanding

Consider the following functions written in Python:

```
def rsel(full_name, mat_string):
    uniq = []
    for c in full_name:
        if c not in uniq:
            uniq.append(c)

    r = []
    i = len(mat_string) // 2
    if i > 0:
        n = int(mat_string[i])
        if n < len(uniq):
            r.append(uniq[n])
            new_full_name = full_name[:n] + full_name[n+1:]
            new_mat_string = mat_string[:n] + mat_string[n+1:]
            r.extend(rsel(new_full_name, new_mat_string))

    return r
```

Consider the variable `my_mat_string` containing the string of all the ten numbers in your matriculation number (e.g. `"0000123456"`), and the variable `my_full_name` containing string of your full name all in lowercase with no spaces. What is the value returned by calling the function `rsel` as shown as follows:

```
rsel(my_full_name, my_mat_string)
```

Section 3: development

The algorithm **globbing** is an algorithm for matching wildcards, which is useful when comparing text strings that may contain wildcard syntax. The most used wildcard is "*" and represent zero or more characters. Examples of use of wildcard search are listed as follows:

- `*foo*` matches any string containing `"foo"`, e.g. `"foo"`, `"fidafoo"`, `"farfoofi"`, `"footer"`;
- `fee*` matches any string that begins with `"fee"`, e.g. `"fee"`, `"feedoo"`;
- `*fae` matches any string that ends with `"fae"`, e.g. `"fae"`, `"fafae"`;
- `doe` (i.e. no wildcards used) matches exactly with the string `"doe"`.

Write an algorithm in Python – `def glob(p, s)` – which takes a string `p` representing the pattern (that may includes zero or more wildcards "*") and another string `s` on which searching for the pattern, and returns `True` if `p` is matched in `s`, otherwise it returns `False`.