Computational Thinking and Programming – A.Y. 2022/2023

Written examination – 20/07/2023

Given name:	
Family name:	
Matriculation number:	
University e-mail:	

Enrolment a. year: [ ] 2022/2023 [ ] 2021/2022 [ ] 2020/2021 [ ] 2019/2020 [ ] 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 *test* an integer number, which of the following Python codes are valid instructions?

- `return test + 5`
- `return test + "abc"`
- `return test, 2`
- `return list(test)`
- `return test.split()`
- `return test.add(4)`

2 – Consider the following Python function:

```
def f(n, s):
    res = n + len(s)
    for c in s:
        if c in "aeiou":
            res = res - 1
    return res
```

What is the value returned by `f(4, "ciao")`?

3 – Write down a small function in Python that takes in input two booleans and returns `True` if both the boleans are `True`, and `False` otherwise.

4 – Describe what are the main components of the Turing machine and what they are used for.

Section 2: understanding

Consider the following functions written in Python:

```
def cntc(given_name):
    d = dict()

    for c in given_name:
        if c not in d:
            d[c] = 1
        else:
            d[c] = d[c] + 1

    res = 0
    for i in d:
        if i in "aeiou":
            res = res + d[i]
        else:
            res = res + (d[i] * 2)

    return res
```

Consider the variable `my_given_name` containing string of your given name in lowercase and without spaces. What is the value returned by calling the function `cntc` as shown as follows:

```
cntc(my_given_name)
```

Section 3: development

The **PageRank** is an algorithm used by Google Search to rank web pages in their search engine results. It works on a directed graph where nodes represent webpages and each directed edge is a link connecting a source webpage to a target one. Each node of the graph has associated a PageRank that measures its relative importance within the graph (the greater, the more important).

In its simplified version, it is computed as follows. It takes in input a directed graph where each node a potential PageRank transfer value to share with other nodes set to 1. Then, the algoritm transfers the such potential value of a given node to the targets of its outbound links, dividing such a value equally among all outbound links. For instance, suppose that page B had a link to pages C and A, page C has a link to page A, and page D has links to all three pages. Thus, page B would transfer half of its existing value (0.5) to page A and the other half (0.5) to page C. Page C would transfer all of its existing value (1) to the only page it links to, A. Since D had three outbound links, it would transfer one third of its existing value, or approximately 0.33, to A, B and C. The sum of all the values that are transferred to a given node is the PageRank of that node – for instance, page A will have a PageRank of approximately 1.83.

Write an algorithm in Python – `def simplified_pr(g)` – which takes in input a directed graph created using the `networkx` library, and returns a dictionary having as many key-value pairs as the number of the nodes in the graph. In particular, each pair has the name of a node as the key and the PageRank of that node as the value. It is possible to use the method `adj[n]` of a graph for getting all the nodes reacheable from a node *n* by following its outbound edges. For instance, considering the example shown above stored as a `DiGraph` in the variable `my_g`, the execution of `my_g.adj["D"]` returns a collection containing the nodes A, B and C.