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

Written examination – 20/06/2022

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 one(s) of the following algorithms presented during the course is recursive?

- Merge sort
- Insertion sort
- Linear search
- Fibonacci (via dynamic programming)
- Line wrap

2 – Consider the following snippet of Python code:

```
def f(s1, s2):
    result = True
    for c in s1:
        result = result and (c in s2)
    return result
```

Which value is returned by calling the function above as follows: `f("riddle", "dialer")`?

3 – Write down a small function in Python that takes in input an integer and returns `True` if it is divisible by 2, otherwise it returns `False`.

4 – Describe, at a general level by introducing its main steps, the algorithm *insertion sort* presented during the course.

Section 2: understanding

Consider the following function written in Python:

```
def cou(mat, n_char):
    n_char_in_mat = n_char % len(mat)
    idx = int(mat[n_char_in_mat])

    mat_l = []
    for c in mat:
        mat_l.append(c)

    result = []
    while len(mat_l) > 0:
        jdx = idx % len(mat_l)
        result.append(mat_l[jdx])
        mat_l = mat_l[:jdx]

    return result
```

Consider the variable `my_mat` containing the string with your full matriculation number (10 digits) and the variable `my_n_char` as the number of alphabetic characters (i.e. excluding spaces) of your given name. What is the value returned by calling the function `cou` as shown as follows:

```
cou(my_mat, my_n_char)
```

Section 3: development

The **odd+11** is an algorithm for finding the anchor day of a given input year. Finding the anchor day of a year is a crucial activity to compute the day of the week of a specific date. The full algorithm, called *Doomsday algorithm*, was devised by John Conway, takes in input a full date (day, month, year) and it is organised in three steps: determination of the anchor day for the century, calculation of the anchor day for the year from the one for the century (the step that should be developed by you in this exercise), and selection of the closest date out of those that always fall on the doomsday.

In particular, the calculation of the number necessary to compute the anchor day of a given input year works as follows:

1. let *T* be the input year's last two digits
2. if *T* is odd, add 11 to *T*.
3. divide *T* by 2.
4. if *T* is odd, add 11 to *T*.
5. *T* becomes equal to $7 - (T \% 7)$.

Write an algorithm in Python – `def odd11(y)` – which takes in input an integer **`y`** representing the last two digits of a given year and returns the number *T* that can be used to find the year's anchor day from the century's anchor day.