2019-09-13 - Pearls of Computer Science - Pearl 001

Study: B-CS Pearls of Computer Science 201700139

Welcome to the digital exam for Pearl 001 Algorithmics.

- You may use 1 A4 sheet with your own notes for this test, as well as a simple calculator
- Scientific or graphical calculators, laptops, mobile phones, books etc. are not allowed. **Put those in your bag now (with the sound switched off)**
- For *technical* questions (concerning the chromebooks, Remindo etc.): raise your mouse
- For *pearl content* question: raise your hand
- Total number of points: 100

Number of questions: 10

You can score a total of 100 points for this exam, you need 55 points to pass the exam.

1
5 pt.

Question 1a

Suppose that you execute the following assignments in Python:

```
region = ["NL", "EU", "OTHER"]
count = [113, 85, 27]
```

region lists the origin region of first-year computer science students enrolled in 2019 (the Netherlands, remaining European countries, and non-European) and *count* lists the (made-up) amount of them.

Write a Python condition (*not* an **if**-statement) that tests whether the sum of the second and third elements of *count* are larger or equal to the first element.

2
5 pt.

Question 1b

Suppose that you execute the following assignments in Python:

```
region = ["NL", "EU", "OTHER"]
count = [113, 85, 27]
```

Write a Python assignment that assigns to a list '*dutch*' the abbreviation and amount of first-year students in computer science from the Netherlands, taken from the lists above.

3
5 pt.

Question 1c

Suppose that you execute the following assignments in Python:

```
region = ["NL", "EU", "OTHER"]
count = [113, 85, 27]
```

Write a sequence of Python assignments that is as short as possible, resulting in a change to *region* after which the amount of students is ordered from lowest to highest number.

(It is *not* correct to assign an entirely new value to *region*; you must modify the existing list.)

4 **Question 2**

Analyse the following Python function

```
1. def compute(data):
2.     result = [ ]
3.     bound = [4, 12]
4.     i = 0
5.     while i < len(data):
6.         if <BLANK1> and <BLANK2>:
7.             <BLANK3>
8.         i = i + 1
9.     return result
```

10 pt. **a.** (a) Change the placeholders BLANK1, BLANK2 and BLANK3 in lines 6 and 7, such that the function call

(1) compute([4, 7, 10, 15]) returns [7, 10],

(2) compute([9, 5, 8, 11]) returns [9, 5, 8, 11],

(3) compute([6, 12, 6]) returns [6, 6].

2 pt. **b.** (b) With these placeholders filled in, what is the return value of compute([8, 2, 4, 5, 8])? -7 [8 8 8]

3 pt. **c.** (c) What does the function *compute* actually do? (Do not give a step-by-step explanation of the execution, but describe the purpose of the function.)

5
10 pt.

Question 3

Recall the algorithm *compute* of Question 2:

```
1. def compute(data):
2.     result = [ ]
3.     bound = [4, 12]
4.     i = 0
5.     while i < len(data):
6.         if <BLANK1> and <BLANK2>:
7.             <BLANK3>
8.         i = i + 1
9.     return result
```

With the placeholders of Question 2a filled in, how many steps does *compute* need in terms of length (say n) of the input list *data*:

- approximately ("in the order of") $log_2(n)$
- approximately $\sqrt{n}$
- approximately n
- approximately n^2

Explain your answer.

6 Question 4

Recall the algorithm *compute* of Question 2:

```
1. def compute(data):
2.     result = [ ]
3.     bound = [4, 12]
4.     i = 0
5.     while i < len(data):
6.         if <BLANK1> and <BLANK2>:
7.             <BLANK3>
8.         i = i + 1
9.     return result
```

5 pt. **a.** Show what changes of *compute* are necessary such that the lower- and upper *bound* changes, depending on an additional input parameter given to *compute*.

5 pt. **b.** With this replacement, does the *order* of steps the algorithm needs to finish its computation change? If so, how; and if not, why not?

7 Question 5a

10 pt.

Consider the following list:

```
[5, 9, 7, -1, 21, 6]
```

Show how bubble sort sorts this list, by writing down the list after every single modification.

8 Question 5b

10 pt.

Consider the following list:

```
[5, 9, 7, -1, 21, 6]
```

Show how merge sort sorts this list, by schematically showing how the list is split and zipped back together.

9
15 pt.

Question 6

Suppose you want to order your entire movie collection by your rating (1 to 5 stars) within a genre, so that all action movies come first, followed by all comedies, then all documentaries and lastly all science fiction movies.

Here are two ways to do this

1. Apply merge sort to your entire movie collection, where one rating is considered to be lower than another, if a movie is lower in the hierarchy (according to: action < comedy < documentary < science fiction) or, if the genre is the same, the movie has a lower rating
2. First distribute all movies into different lists corresponding to the four different genres, then apply merge sort to each genre, and then put the lists back together.

Which of these two solutions is faster, assuming that the amount of movies is evenly distributed over the genres (there are as many comedies as there are documentaries etc.)? Explain your answer by analyzing the approximate number of necessary steps to finish the respective computations of each solution.

10

Question 7

You have recently moved to a new flat and winter is coming. To prepare yourself for the cold season, you want to unpack your unordered box of gloves, and sort them onto your glove shelf. With incredible foresight you have made sure to always buy the same types of gloves, so that each pair of gloves is indistinguishable from another one -- Gloves can still clearly be separated into "left" and "right" glove. Moreover, you are a very careful person, so you know there are as many left gloves as there are right gloves.

10 pt. **a.** (a) Write an algorithm in natural language with unambiguous, numbered instructions, that places pairs consisting of a "left" and a "right" glove onto the shelf. Your instructions may only involve one or two gloves at a time, never an arbitrary set of them. You may assume that you have additional empty boxes at your disposal, which may help you in your sorting efforts. **Do not try to give an answer in Python!**

5 pt. **b.** (b) Now write an algorithm ***def*** *glovesort(data)* in Python, that:

- resembles your pseudo-code algorithm *as closely as possible*,

- takes as input a list *data* consisting only of the letters 'L' and 'R' (indicating left and right). You may assume *data* contains nothing else but an equal amount of L and R, but their order is completely arbitrary,

- outputs a newly ordered list *result* of the form [L, R, L, R, L, R ...].

Thank you, your exam has been saved. You will be notified about your grade after your answers were checked by the correction team