PwD Coursework II: **Problem solving and scalability analysis**

v. 2 of March 10th 2020, amended text is in blue.

This is the second coursework of this module. This documents is implicitly extended with the general college rules and prohibitions regarding coursework which were set forth in the text of the first coursework document regarding, e.g., late submission and plagiarism. In what follow students will be assumed to be acquainted with the text of Coursework I.

The goal of this coursework is to make you apply the concepts and general methods seen in class and understand the scalability of your code.
This assignment requires you to upload a short Python source code and an essay/technical annex that describes

i) your analysis of the problems described below,
ii) your algorithmic solution and the Python code for it and
iii) the scalability analysis of your code, tested against random instances of the problem.

Work is organized in phases, as described below.

Phase 1: MyHealthcare device: Vital signs simulator (20 marks)

MyHealthcare device is a wearable device for collecting vital sign data.
Develop a Python function to simulate the **MyHealthcare device** that generates data for "n" vital sign records (e.g. 1000, 2000 etc.) of a person.
The data should be generated according to the rules as presented in the following table.
Measures that are out of the normal range are considered abnormal e.g. for temperature normal values are: 37 and 38 and abnormal values are 36 and 39.

	Temperature	Heart rate	Pulse	Blood pressure	Respiratory rate	Oxygen saturation	pH
Values to generate	36-39 (int)	55-100 (int)	55-100 (int)	120-121 (int)	11-17 (int)	93-100 (int)	7.1-7.6 (float, 1 decimal)
Normal rates	37-38	60-99	60-99	120	12-16	95-100	7.3-7.5
Abnormal rates	36, 39	55-59, 100	55-59, 100	121	11,17	93,94	7.1, 7.2, 7.6

The table below shows example data for three records, n=3.
Each data row is called record and consists of 8 values (namely as timestamp, temperature, heart rate, pulse, blood pressure, respiratory rate, oxygen saturation and ph). In this example, abnormal values are highlighted with red font.
Notice that here "ts" stays for timestamp (feel free to use your own timestamp or a Python timestamp).

ts	temp	hr	pulse	bloodpr	resrate	oxsat	ph
101	37	61	88	120	**11**	95	7.3
102	**36**	77	75	**121**	13	100	7.4
103	37	76	**54**	120	14	**94**	**7**.2
…							

Notes: To ensure consistency of results please use "seed(109)".
Generate data for thousand records, n = 1000.
Name your function ***myHealthcare(…)***

Phase 2: Run analytics (30 marks)

Develop a Python function for each of the following analytics:

a) Find abnormal values for pulse or blood pressure.
   - Select a small sample e.g. 50 records and count the instances where a vital sign was out of the normal range for a selected value.
   - Return selected values for each timestamp.

   An example output for pulse with 3 abnormal values could be
   [pulse, 3, [[105,56], [109,57], [125,59]]]
   or
   {" abnormal_pulse_count": 3, "abnormal_values": [[105,56], [109,57], [125,59]]},
   where [105,56] is [timestamp,value].
   Feel free to create your own data structure.

b) Present a frequency histogram of pulse rates.
   - Select a small sample e.g. 50 records, find the frequency for pulse rate values.
   - An example output could be: [[55,2],[56,6],[57,4],[59,12],…]

c) Plot the results for 2a and 2b and briefly discuss your observations.
d) What is the computational cost (as a function of n) of your solution?

Please present diagrams and discussions in the report.
Please name your functions, e.g., abnormalSignAnalytics(…), frequencyAnalytics(…).

Phase 3: Search for heart rates using the HealthAnalyzer (30 marks)

Develop a function called HealthAnalyzer. HealthAnalyzer provides a query mechanism to search for a particular sign value, for example it could search for records where the pulse value is 56.

a) Design a solution (including one or more algoritms) to search for a particular pulse rate value (e.g. 56), the algorithm should return a multidimensional list with all the records associated with this value (ts, temp, hr, pulse, bloodpr, resrate, oxsat). Keep in mind that a value might exist more than 1 time.
For example, when searching for value 56, the following 3 records are selected.
[121, 37, 61, **56**, 120, 11, 95, 7.3],
[126, 38, 62, **56**, 120, 11, 95, 7.3],
[131, 37, 62, **56**, 120, 11, 94, 7.2]

For this example, the output could be a list such as: [[121, 37, 61, 56, 120, 11, 95, 7.3], [126, 38, 62, 56, 120, 11, 95, 7.3], [131, 37, 62, 56, 120, 11, 94, 7.2]].

b) What is the complexity of your solution?

c) Plot the heart rate values for records having pulse rate 56.

Please present diagrams and discussions in the report.
Name the function: healthAnalyzer (…)

Phase 4: Testing scalability of your algorithm (20 marks)

Benchmark *all phases* of the coursework, simulating n = 1000, 2500, 5000, 7500 and10000 records from phase 1 (MyHealthcare device). This includes algorithms for: Data generation (from phase 1), analytics (from phase 2) and health analyser (from phase 3). Provide your benchmark study in the report with a clear definition of the computational complexities of the algorithms.

a) Measure the running time and plot the results for different n values.
b) Present diagrams and discussions in the report.

Name the function: benchmarking(myHealthcare(…))

General guidelines:

- Provide comments in the code to explain functionality.
- Please follow the instructions on your Moodle page for time and mode of submission of the solution.

Please note:

- We expect you to provide more than one solution and compare your findings in terms of "what is the better algorithm to use?" (for example, why you might use linear search or binary search or interpolation search for implementing your algorithms, what are the trade-offs?).

- Extra marks will be awarded for "home-made functions" with clear description on algorithmic complexity (in comparison to the build in Python functions with sometimes unknown low level detail and complexity).

Submission: Please upload (a) the Python source codes of your solution and (b) a technical annex (report).
About the technical annex: please use your judgement on the right amount of data and length of presentation for a technical description of your solution.
In these instructors' opinion, two pages should suffice.
Please submit your technical annex in a PDF format.
Please use an easy-to-read style similar to that of this document (Times New Roman font or similar, size 12, 1.15 line spacing or higher, justified alignment).

Deadlines: please refer to Moodle.

Citations & Plagiarism: please refer to the detailed presentation in the text of Coursework 1