COMP101
Introduction to Programming 2024-25
Assignment-04

Issue Date:	**Thursday 31st October 2024**
Submission Date:	**Thursday 07th November (17:00)**

1. Overview

a) Summary:

This assignment is worth 13% of the total marks for COMP101.
Plagiarism and collusion guidelines apply throughout.

b) Guidance:

Assessment is based on design, clarity, accuracy and nested loops and selection.

- **Do not modularise the code, do not use break/continue**
- **No tests or design document is needed**

c) Submission:

a) Submit **one** .py file with filename in format of:
familyName_givenName-CA04.py

Within the code, the first lines should be comment lines as follows:
#familyName_givenName Month and Year of coding filename.py
#Smith_John Nov 2023 CA-04.py

Submit your documents electronically via the department submission server:
https://sam.csc.liv.ac.uk/COMP/Submissions.pl

Earlier submission is possible, but any submission after the deadline attracts the standard lateness penalties, see:
http://www.csc.liv.ac.uk/department/regulations/practical.html

2. Assessment Information

Assignment Number	04 (of 07)
Weighting	13%
Assignment Circulated	See front page
Deadline	See front page
Submission Mode	e-submission
Learning outcome assessed	1, 2, 4, 5, 7
Purpose of assessment Marking criteria	Using sequence, selection and iteration constructs to control I/O of strings and numbers in successful calculations for a given problem, using import as required. Total marks over seven questions as a percentage
Submission necessary in order to satisfy module requirements?	Yes Assignments are not marked anonymously
Late Submission Penalty	Standard UoL Policy.

3. Problem Specification:

3.1: Background
Students (176 in total) enrolled on a module are expected to submit an assignment.
The assignment has two parts to be submitted for assessment:
a) Code
b) Tests
Both parts are submitted at the same time on the same day and carry equal weight.

3.2: Submissions
We don’t know how many students will make a submission.
We do know how many parts have to be graded when there is a submission to grade.

3:3: Grading
Students could submit nothing at all, could submit only one part (code or tests) or could submit both parts (code and tests).
The grade range for both parts is 0 to 100 inclusive.

You can code for a submission that, for each part submitted, will score minimum 1 mark up to maximum 100 marks (0 is used to indicate non-submission of a part).

Students who submit only one part (it doesn’t matter which one) will receive a grade (1 to 100) for that part and will receive a grade = 0 for the other part not submitted.
If a student does not submit both parts, then both parts = 0.

3.4: Late
In addition, submissions are accepted up to 2 days late with deduction of 5 marks (not percentage of grade) per day late.

Processing:
Input:
Only accept valid integer input and prompt for re-input if invalid.
Only proceed with processing when input is validated.
There is no requirement to 'time-out' the user after, say, three attempts etc. Sone or later they will input something valid.
Capture:
a) Student id (auto-generate this if you can), Code: Integer grade 0 to 100, Tests: Integer grade 0 to 100, Late: Integer 0, 1 or 2

Process:
raw grade = average of code grade + test grade
final grade = (raw grade) – late penalties

Output:
With every student processed:
e.g. first student submits both parts and on time
id: 1, code: 40, tests: 60, raw: 50, late: 0, final: 50
e.g. next student submits one part only and 2 days late
id: 2, code: 0, tests: 80, raw: 40, late: 0, final: 30

It is expected that you recognise the above example output could be improved for user-facing or user-friendly – although they are accurate.
Improved output format is at your own design.

Mark scheme

Analysis and Design	40%
Implementation	35%
Validation/nesting	25%

The mark scheme looks for:

- Use of nested constructs (for, while, if)
- Validation on integer input
- Output that is understandable to the user
- Code clear and readable supported with appropriate in-line commentary

Exclusions:

No menu

No test table

No break, continue or pass: Assessment looks for good programming in controlling nested loops and not a brute-force attack on it to make it stop, i.e. you need to show you understand how to program loops and make them stop naturally without forcing it.