COMP1038 Coursework 1 – Students grades management

Introduction

This is the first COMP1038 Coursework. It is worth **30% of the module mark**. It requires you to write an interactive, menu-driven program that acts as a student grades management program. The deadline for this exercise is **18:00 on Thursday 21st of November 2024**.

Read the entire document before beginning the exercise.

If you have any questions about this exercise, please ask in the Q&A forum on Moodle, after a tutorial session, or during the advertised office hours. Do not post your program or parts of your program to Moodle as you are not allowed to share your coursework programs with other students. If any questions require this exercise to be clarified, then this document will be updated and everyone will be notified via Moodle.

Submission

You must submit a single C source code file containing all your code for this exercise. This file must be called **student_grades.c** and must not require any other files outside of the standard C headers which are always available. The first line of the file should be a comment which contains your student ID number, username, and full name, of the form:
// 6512345 zy12345 Joe Blogs

The file must compile without warnings or errors when I use the command
gcc -std=c99 -lm student_grades.c -o student_grade
This command will be run on our Linux server cs-linux. If it does not compile, for any reason, then you will lose all the marks for testing (common reasons in the past have been submitting a file with the wrong filename or developing your solution on your personal computer without having tested it on our Linux server). If the file compiles but has warnings, then you will lose some marks for not correcting the warnings.

Before you submit your source code file, you must complete the **Coursework submission coversheet** on Moodle.

The completed source code file should be uploaded to the Coursework 1 Submission link on the COMP1038 Moodle page. You may submit as many times as you wish before the deadline (the last submission before the deadline will be used). After the

deadline has passed, if you have already submitted your exercise then you will not be able to submit again. If you have not already submitted, then you will be allowed to submit **once**.

Late submissions: Late submissions will lose 5 percentage points **per hour**, rounded up to the next whole hour. This is to better represent the large benefit a small amount of extra time can give at the end of a programming exercise. No late submissions will be accepted more than 24 hours after the exercise deadline. If you have extenuating circumstances, you should file them before the deadline.

Plagiarism

You should complete this coursework on your own. Anyone suspected of plagiarism will be investigated and punished in accordance with the university policy on plagiarism (see your student handbook and the University Quality Manual). This may include a mark of zero for this coursework.

You should write the source code required for this assignment yourself. If you use code from other sources (books, web pages, etc), you should use comments to acknowledge this (and marks will be heavily adjusted down accordingly). *The only exception to this is the example programs given in lectures and tutorials; you may use them, with or without modification, without penalty.*

You must not copy or share source code with other students. You must not work together on your solution. You can informally talk about higher-level ideas but not to a level of detail that would allow you all to create the same source code.

Remember, it is quite easy for experienced lecturers to spot plagiarism in source code. If you are having problems, you should ask questions rather than plagiarize. If you are not able to complete the exercise, then you should still submit your incomplete program as that will still get you some of the marks for the parts you have done (but make sure your incomplete solution compiles and partially runs!).

Task

A simple student grades management software was developed by a computer science student, and the executable file is available on Moodle. While the software functions

correctly with valid user input, it is not fully satisfactory as it does not handle incorrect input effectively. Here are a few examples: 1) When the program prompts the user to choose an option, it may crash if non-digit characters are entered. 2) The program does not validate whether the student names and grades are entered correctly.

Your task is to write a program that implements the same student grades management system but with the ability to handle invalid user input. The requirements for handling invalid input are as follows:

1) Student names must be unique. If a newly entered name matches an existing one, it should be considered invalid and discarded.

2) Grades must be within the range of [0, 100].

3) A valid name must begin and end with an English alphabetic character. Student names can be a maximum of 20 characters long, including spaces.

4) In the event of invalid input, the program should allow the user to re-enter the information without displaying any message.

You may assume that the maximum number of students is 1000.

Marking

The marking scheme will be as follows:

- *Tests (90%):* Your program should correctly implement the task requirements. A number of tests will be run against your program with different input data designed to test if this is the case for each individual requirement. The tests themselves are secret but general examples of the tests might be:
    - Does the program work with typical valid input?
    - Does the program correctly deal with input around boundary values?
    - Does the program correctly deal with invalid input values?
    - Does the program work with a proper user interface, i.e. exactly same as the given executable file?

As noted in the submission section, **if your program does not compile then you will lose all testing marks**.

- *Source code formatting (10%):* Your program should be correctly formatted and easy to understand by a competent C programmer. This includes, but is not

limited to, indentation, bracketing, variable/function naming, and use of comments. See the textbook for examples of correctly formatting programs.

END