ProgSD Lab Exam (October 2023) general feedback

- The lab exam was an open book.
- In general, most of you did well because the codes were marked manually. If we had used fully automatic marking, many would have failed. Please take care to read instructions very carefully!
- Python average 16.2/25.
- Java average 11.2/25
- For both questions, we would run your code. If the output is correct, you will get the full marks. Partial marks were given to some codes when the output was incorrect, but we could understand the intention.

We are very happy that this year's collusion level was very low compared to last year's.

Python: We already covered these topics. Many students lost marks due to submitting incomplete or partial solutions or late submissions. Note that all students were given the full 1 mark for code style this year.

It was about creating a program that should display a screen as per the screenshot requirements. When it was run, if the windows appeared as in the example provided (the dimensions were not important), you would get **5 marks**. However, if the name of the app or any of the text boxes or display buttons were missing, you would get marks deducted.

The marker would check the code if a database called **LibraryDB** was created. If it is called differently, lose 0.25 marks. The marker would enter the book's ISBN, title, author, and year of publication and save the valid book details into a database named LibraryDB by clicking the **"Save"** button. A suitable message should appear when book details are saved. **(4 marks).** You lose 0.5 if there is no pop-up message.

The marker would then clear **ALL** the windows (as you can see from the screenshots, there are many, not just the first four data entry ones) when the **"Clear"** button is clicked, and you get **2 marks.**
The marker would click the **"Display Books"** button**, and** the system would display all book details saved in the database. (**3 marks).** Unless this was really impossible to see, the way these were displayed did not matter. As we said, marking was very generous.

The marker would then check the validations: 1) The system should ensure that no duplicate **ISBNs** are stored. The system should display a suitable message if a book with the entered ISBN already exists (**1 mark).** 2) Ensure that the entered **year** is **not in the future** AND not earlier than **1450** (the approximate year of the first printed book) (**1 mark).** 3) Ensure that neither the **title** nor the **author** fields are empty. If any fields are empty, the system should display a suitable message. (**1 mark).** A student would lose a fraction of the mark for each of these validations when no pop validation message is provided.

For the last Python task, you were to simulate the interaction with ChatGPT using a local function. The system was to connect to the 'LibraryDB' to retrieve real-time data to answer the user's queries. The virtual librarian was to respond to queries such as:"**Recommend me a book by** [author name]." **(3.5 marks)** "**When was published** [book title]." **(3.5 marks). To mark this task, the marker would look inside the code to see HOW you wrote the queries and use those specific terms to test the system. For queries that did not work, many students received between a fraction of the marks (0.25, 0.5, 1, or even 1.5 marks) depending on the code provided.**

Java: We used automated tests to check your code for all tasks. If a test did not pass, we manually inspected the reason for the failure, in order to assign marks correctly. All students were given the full 1 mark for code style in Java as well.

Task 2a: Representing individual seats

Many students got full marks on this task, or at least partial marks (as indicated below)

Task 2a.1: full marks were given for a version of **Seat.java** that had appropriate fields for row, seat number, seat type, and availability; a constructor that set them appropriately; and appropriate access modifiers (private or protected) on the fields. Partial marks were given not all of that held. 1 mark out of 4 for an empty class.

Task 2a.2: full marks if the row and seat number are validated correctly and an exception thrown correctly. Partial marks if validation is attempted but the checks are wrong or the exception thrown incorrectly.

Task 2a.3: full marks only if all necessary getters and **one** setter were implemented – partial marks if getters were missing and/or extra setters provided.

Task 2b: Representing a venue

Fewer students submitted this class, however many did still get full marks.

Task 2b.1, full marks if the **Venue.java** class stored the seats in a sensible data structure, and also had a constructor that created the seats based on a String parameter. Partial marks if the constructor did not process the string correctly. Many students created the class but not the constructor, and many stored only a single Seat rather than a collection of them.

Task 2b.2: full marks if **getSeat** was implemented in a way that worked with the seat representation chosen. Partial marks if method is included but is not functional.

Task 2b.3: full marks if **printDetails()** is implemented in some reasonable fashion; partial marks if method specified but not functional.

Task 2c: Representing an event

Not many students submitted a full solution to this part

Task 2c.1: Full marks if the **Event.java** class stored a venue and a price structure and sets them somehow; partial marks for an incomplete submission

Task 2c.2: Full marks if a reasonable implementation of **reserveSeat** is provided (finding the seats, reserving them, returning the total price); partial marks if attempt is made but not fully functional; 1/4 only for method header with little or no body

Task 2c.3: Full marks if **returnSeat** checks validity, throws exceptions as appropriate, and updates availability if needed; partial marks for incomplete attempts.