PROGRAMMING 512

RICHFIELD
richfield.ac.za

Name and Surname: _____

Student ITS No: _____

Qualification: _____ **Year of Study:** _____ **Semester:** _____

Assignment due date: _____ **Date submitted:** _____

QUESTION	EXAMINER MARKS	MODERATOR MARKS	REMARKS

ASSIGNMENT INSTRUCTIONS

Please tick each box to confirm completion.

Use Times New Roman font, size 12, with 1.5 line spacing throughout the document.
Apply Harvard Referencing Style for all citations and references.

For essay-style assignments, please include the following sections:

Table of Contents
Introduction
Main Body (with relevant subheadings)
Conclusion
References

Submit the assignment in PDF format on Moodle.
Use the specified cover page provided.
Include a signed declaration of originality.

DECLARATION OF ORIGINALITY:
I hereby declare that this assignment is my own work and has not been copied from any other source except where due acknowledgment is made. I affirm that all sources used have been properly cited and that this submission complies with the institution's policies on academic integrity and plagiarism.

Student Signature: _____ **Date:** _____

Question 1 (30 marks)

1.1 Compare Lists, Tuples, Dictionaries, and Sets by creating a table that shows how they differ based on the following characteristics:

a. Mutability
b. Ordering
c. Duplicate Elements
d. Indexing Method
e. Syntax Creation **(10 marks)**

1.2 Analyse the relationship between abstraction and encapsulation in object-oriented design. Using a practical example, demonstrate how these two concepts work together to create robust, maintainable software. **(10 marks)**

1.3 Explain how databases are used in Python using SQLite. In your answer, include:

- What SQLite is and why it is commonly used in Python applications.
- The typical steps involved in interacting with an SQLite database using Python, including code-related keywords or functions.
- One advantage and one disadvantage of using SQLite in a real-world application.

(10 marks)

Question 2 (30 marks)

Create a Tic Tac Toe game in Python using object-oriented programming principles. The game should have a console-based interface that displays the board grid and prompts users for input (see example interface below).

```
=== Welcome to Tic Tac Toe ===
Select game mode:
1. Player vs Player
2. Player vs Computer
Enter 1 or 2:  1
Enter Player 1 name (X):  Amy
Enter Player 2 name (O):  John

    1   2   3
  +---+---+---+
A |   |   |   |
  +---+---+---+
B |   |   |   |
  +---+---+---+
C |   |   |   |
  +---+---+---+
Amy's turn (X)
Enter position (e.g. A1):
```

Part 1: Game Setup and Interface **(10 marks)**

When the program starts, it should:

- Display a welcome message
- Ask the user to choose between **Player vs Player** and **Player vs Computer**
- Ask for player name(s) and assign Player 1: X, Player 2 or Computer: O
- Display a grid-based board using rows labelled A–C and columns labelled 1–3
- Allow players to make moves by entering positions like A1, B2, etc.
- Display the updated board after each move

Implementation:

- Use a **Board class** that manages the game board, displays the grid, and handles move validation.
- Use a **Player class** to store player names, symbols (X or O), and whether a player is human or computer.

Part 2: Game Logic **(10 marks)**

The program must:

- Alternate turns between players
- Detect and announce a win or a draw
- In **Player vs Computer** mode, the computer should:
  - Try to win if possible
  - Block the opponent from winning
  - Take the centre if available
  - Prefer corners next
  - Otherwise, pick a random available space.
- After the game ends, display a message showing the winner or if it was a draw

Implementation:

- Use a **Game class** that controls game flow, including turn switching, mode selection, input handling, and AI logic.
- The **Board class** should include methods to check for wins or draws.

Part 3: Scoreboard and Replay **(10 marks)**

After each match:

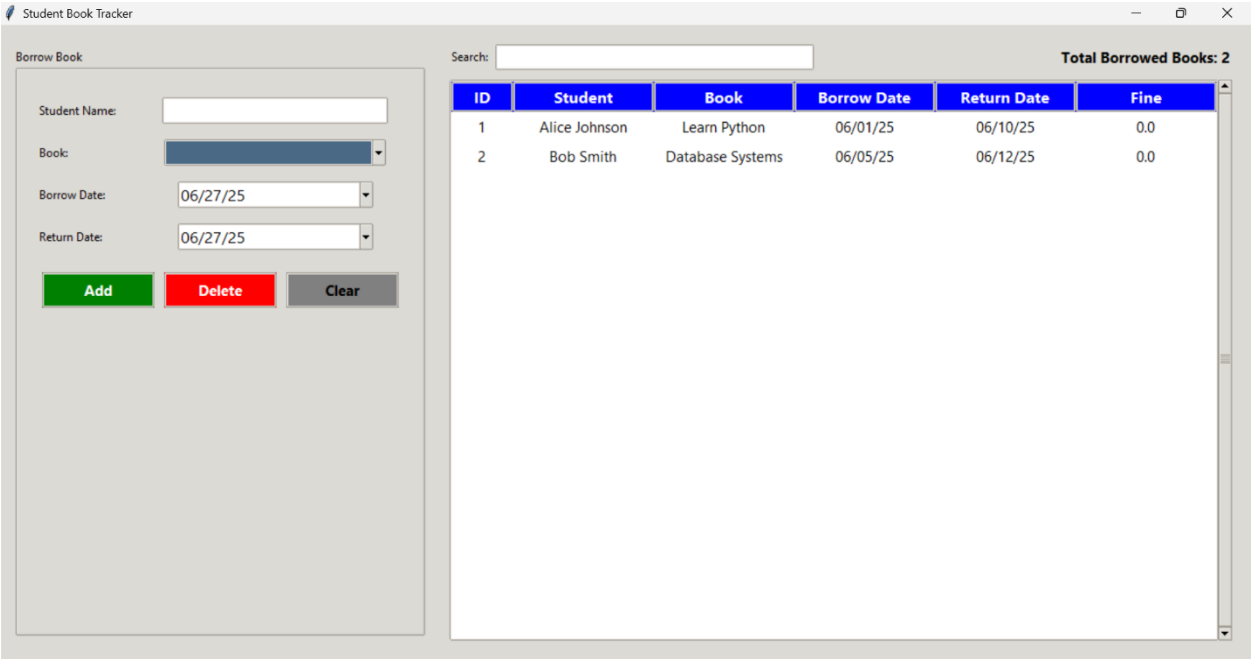
- Update and display a scoreboard that keeps track of Player X wins, Player O wins, and draws
- Ask the user if they want to play again
- If yes, reset the board and play a new round with the same players
- If no, display a goodbye message and end the program

Implementation:

- The **Game class** should handle scoreboard tracking and replay functionality.

Question 3 (40 marks)

Create a Tkinter-based library management application that tracks borrowed books, monitors due dates, and automatically calculates late return fees. Your application should provide an intuitive interface for managing the library's book lending system (see example interface below).



Part 1: Database Setup (15 marks)

Create a database file named library.db with two tables:

1. **books**
   - id - primary key, autoincrement
   - title - book title (text, not null)
   - quantity - number of available copies (integer, not null)
2. **borrowed_books**
   - id - primary key, autoincrement
   - student_name - name of the student (text, not null)
   - book_id - references the id from the books table
   - borrow_date - text format (e.g. “06/20/25”)
   - return_date - text format
   - fine - real number (default = 0)

You should also insert:

- At least 5 sample books in the books table.
- A few sample records in borrowed_books to help you test the table display later.

<u>Part 2</u>: Borrow Book Form **(10 marks)**

All widgets must be placed neatly inside a LabelFrame and include the following:

1. A text entry field for the Student Name.
2. A combobox for selecting a Book. This list should only include books where the quantity is greater than 0.
3. Date Pickers:
   - A DateEntry for Borrow Date (default should be today's date).
   - A DateEntry for Return Date (also defaulted to today).
   - The Borrow Date cannot be in the past, and the Return Date must not be before the Borrow Date.
4. Action Buttons (placed below the input fields):
   a. Add (green button):
      - Adds the new record to the database.
      - Decreases the quantity of the selected book by 1.
      - Calculate the fine if the return date has passed (R5 per day).
      - Refreshes both the table and the book dropdown list.
   b. Delete (red button):
      - Prompts a Yes/No confirmation before deleting.
      - Increase the book quantity by 1 if confirmed.
      - Refreshes the table and dropdown list.
   c. Clear (grey button):
      - Clear the text entry and reset both dates to today.

<u>Part 3</u>: Borrowed Books Table and Search **(15 marks)**

1. A search bar (label + entry box) at the top. It should filter the table in real-time as the user types, matching either the student's name or the book title (case-insensitive).
2. A label that shows the total number of records currently displayed.
3. Below the search area, add a Treeview table with these columns: ID (narrower column), Student, Book, Borrow Date, Return Date and Fine.
4. The table should have a blue header with a bold white font, highlight rows in red where the fine is greater than 0 and include a vertical scrollbar correctly aligned next to the table.