Testing Exam 1 - Wordle

> Any question asked whose answer is directly stated in this document will result in a **-1 point penalty**.

Introduction

Wordle is a word puzzle game where players attempt to guess a five-letter word within six attempts. After each guess, the game provides feedback using colors:

- **Green**: The letter is correct and in the right position.
- **Yellow**: The letter is in the word but in the wrong position.
- **Gray**: The letter is not in the word.

This lab focuses on implementing Wordle while practicing unit testing and various testing techniques. You'll build the core game logic and ensure its correctness through comprehensive testing.

We recommend adopting a **test-first approach** (not necessarily TDD).

Task Requirements

> *(Bonus points will be awarded for features that are correctly implemented and tested.)*

Basic Requirements

- Implement word validation (5 letters, alphabetic characters only).
- Implement guess-checking logic (correct letter/position feedback).
- Handle basic game state (remaining attempts, game-over conditions).
- Minimum test coverage: **70%**.

Intermediate Requirements *(+2 points)*

- Implement a dictionary of valid words.
- Handle duplicate letters correctly.
- Maintain game statistics (wins, streak, average attempts).
- Minimum test coverage: **80%**.

Advanced Requirements *(+4 points)*

- Implement multiple game modes (e.g., timed mode, practice mode).
- Support dynamic word lengths.
- Develop a score calculation system.
- Minimum test coverage: **90%**.

Evaluation Criteria

Testing Strategy Quality (75%)

Your testing approach will be evaluated based on the following principles:

- The **five main rules of unit testing** are followed:
- 🎯 **Test one behavior at a time.**
- 🔌 **Do not test dependencies.**
- ⚡ **No long-running tests.**
- 🎲 **Tests must be deterministic.**
- 🏝️ **Tests must be isolated.**
- **Dependency Injection (DI) is used effectively.**
- **Test doubles (mocks, stubs, fakes) are used efficiently.**
- **Test coverage meets or exceeds 70%.**

Implementation Quality (25%)

- Your program meets the specified requirements.
- Edge cases and error handling are tested.
- Git is used for version control, with an iterative development approach.

Implementation Notes

- You may use **any programming language**.

- The implementation can be a **console, desktop, web, or mobile application**.
- Follow **object-oriented or functional programming** principles.
- The code must be **fully functional and executable**.
- Include documentation explaining how to **run your tests and application**.

Example Test Structure:

```
WordValidator Tests:
├── validateWord()
│   ├── Rejects words shorter than 5 letters
│   ├── // Test 2
│   ├── // Test 3
│   └── // Test 4
│
GameLogic Tests:
├── checkGuess()
│   ├── Identifies exact matches
│   ├── // Test 2
│   ├── // Test 3
│   └── // Test 4
│
GameState Tests:
├── makeGuess()
│   ├── Updates remaining attempts
│   ├── // Test 2
│   └── // Test 3
```

Submission Requirements

> **Note:** Be fully prepared before presenting. **The 5-minute timer will start as soon as you are called.**

1. **GitHub Repository**

- The repository **link must be uploaded to Moodle** by the last session day at **2:00 PM**.
- Your repository must include:
  - **Complete source code**
  - **Test suites**
  - **Project documentation in a `README.md` file**

2. **Individual Presentation (5 minutes max - Timed)**

- Live execution of **all tests**, demonstrating passing and failing tests.
- Display of **test coverage metrics**.
- Working **demo** of the application.
- Brief explanation of the **code architecture**.

3. **Technical Defense**

- Be prepared to explain **any part of your code**.
- Justify **technical decisions**.
- Explain your **testing strategy**.

Important Notes

- **Strict Deadlines:**
  - **Late submissions will not be accepted.**
  - **Incomplete presentations will result in a failing grade.**
- **During the Exam:**
  - Any question asked whose answer is directly stated in this document will result in a **-1 point penalty**.
  - **Read the entire document carefully before starting.**
  - Make sure you **understand all requirements before asking questions**.
- **Technical Requirements:**
  - **Coverage requirements must be met.**
  - **The application must be fully functional during the demo.**

Penalties

- **-10 points** if your code does not run.
- **-1 point per minute** past the submission deadline.