Week 4 Progress Report

- Dates: 9/29 - 10/5

Week 4 Goals Completed (3/3)

✅ As a developer, I want to generate tests for edge cases and error handling to ensure the logic works as expected.
✅ As a developer, I want to continue optimizing and cleaning up the code to ease future development.
✅ As a developer, I want to add code comments and documentation so my teammates can easily understand my code.

Additional Work Completed

✅ Added error handling for `Row`, `Board`, and `LinkedList` classes

Summary of changes under `src/` and `tests/`

- Files changed:
  - `src\game\board.py`
  - `src\game\row.py`
  - `src\starter_code\tetris_code_explained.py`
  - `src\starter_code\tetris_ver1.py`
  - `src\utils\linked_list.py`
  - `tests\test_board_core.py`
  - `tests\test_board_edge_cases.py` (new)
  - `tests\test_linked_list_core.py`
  - `tests\test_linked_list_edge_cases.py` (new)
  - `tests\test_row_core.py`

Major Changes

1. `board.py`
    - Purpose
        - Encapsulates the playing field and provides board-level operations (cell access, clearing, full-line detection and removal) using `Row` bitboards stored in a `LinkedList`.
    - Changes
        - Constructor / API
            - `Board` now requires a `row_factory` callable (zero-arg factory that returns a `Row`-like object). Passing a non-callable raises `TypeError`.
            - Exposes `.height` and `.width` properties (backing attributes `__height` and `__width`).

2. `linked_list.py`

- Purpose
  - Lightweight singly linked-list implementation used to store `Row` objects in `Board` (one node per row) and support efficient top insertion and mid-list deletion.
- Changes
  - Validation & exceptions
    - Insert/append operations now validate values and raise `ValueError` when attempting to append/insert `None`.
    - Index checks (`_check_index()`) raise `IndexError` for out-of-range indices; `get_node_at()` and `delete_node()` propagate these exceptions rather than printing errors or returning `None`.
  - Length tracking

3. `row.py`
    - Purpose
        - Represents a single board row using a compact bitmask (`__bits`) and a per-cell color mapping; provides bit operations and fullness checks.
    - Changes
        - Input validation: `Row(width)` raises `ValueError` for non-positive widths.
        - Column index checks: `get_bit()` , `set_bit()` , and `get_color()` validate column indices and raise `IndexError` for out-of-range accesses.
        - Core operations: `is_full()` , `clear_row()` , `set_bit()` , `get_bit()` , and `get_color()` implemented with clear semantics and internal color tracking.
    - Why
        - Prevent invalid row construction and out-of-bounds bit access. The explicit

4. Tests (`tests/`)

- Purpose
  - Provide both small, fast core tests and more comprehensive edge-case tests exercising the new error behavior and board semantics.
- Changes
  - Renamed existing files:
    - `test_board.py` -> `test_board_core.py`
    - `test_row.py` -> `test_row_core.py`
    - `test_linked_list.py` -> `test_linked_list_core.py`
  - New files for testing edge cases:
    - `test_board_edge_cases.py`
    - `test_row_edge_cases.py`

5. Starter code (`tetris_ver1.py`, `tetris_code_explained.py`)

- Purpose
  - Provide runnable example/starter scripts that exercise the `Board` API and demonstrate usage.
- Changes
  - Updated initialization to construct a `Board` with an explicit factory (example: `Board(lambda: Row(constants.WIDTH))`).
  - Replaced legacy helper calls with the new API: use `GameBoard.height` / `GameBoard.width` instead of older `get_height()` / `get_width()` methods.
- Why
  - Keep examples in sync with the library API so new contributors can run the

6. Misc / Notes

- Tests and CI
  - After these changes, the repository unit test suite was executed and all tests passed (41 tests).
- Rationale summary
  - The week focused on hardening the board/row/linked-list abstractions with proper input validation and explicit failure modes, simplifying test expectations and making the codebase easier to reason about and extend.