INFO0004-2: Othello game engine

March 11, 2025

1 Introduction

In this assignment, you will design and implement an othello game engine.

Othello is a strategy board game whose simple rules are explained here.

2 Interface

Your game engine will provide the following public interface, in an `othello.h` header file:

```
#include <set>
#include <vector>
#include <utility>
#include <stdexcept>

class Othello {
    public:
        /* used to represent the (row, column) position on the board
           pair.first is row (0-7), pair.second is column (0-7)*/
        typedef std::pair<int, int> POSITION;
        enum Colour {EMPTY, WHITE, BLACK};

        /* creates a new, default game */
        Othello();

        /* board must represent a 2-dimensional "array" of 8-by-8
           player must be Colour::WHITE or Colour::BLACK, otherwise invalid_argument is thrown
           throws length_error if board is not at least 8-by-8 */
        Othello(const std::vector<std::vector<Colour>>& board, Colour player);

        /* returns colour of player whose turn it is; Colour::EMPTY if nobody can play */
        Colour getActivePlayer();

        bool isGameOver();

        /* returns Colour::EMPTY if game is not finished */
        Colour getWinner();

        /* returns all available moves to the active player; empty set if game is finished */
        std::set<POSITION> getMoves();
```

```
        /* plays disc of active player at pos, and updates game status;
           returns true on success, false if move not allowed */
        bool play(POSITION pos);

        /* returns colour at pos */
        Colour value(POSITION pos);
};
```

You are free to design the *private* part of this class as you see fit, but you must scrupulously respect the public interface. The `othello.h` file *must only contain the strict minimum code required* to support the use of this interface.

All implementations must be provided in an `othello.cpp` file.

You can consult the Standard C++ Library Reference for further documentation about the STL data structures and algorithms, at `http://cppreference.com/`.

3 Remarks

3.1 Respect the interface

Submission must scrupulously **follow the interface** defined above. You can of course define auxiliary functions as you see fit, but these should be properly hidden from the users of the interface.

3.2 Readability

Your code must be readable:

- Make the organisation of your code as obvious as possible. Remember you can create as many auxiliary and private functions as you see fit.
- Use descriptive names for functions.
- Complement your self-documenting code with comments, where appropriate.
- Choose a coding convention, and stick to it. *Consistency* is key.

3.3 Robustness

Your code must be robust. The `const` keyword must be used correctly, sensitive variables must be correctly protected, memory must be managed appropriately, the program must run to completion **without crash**.

3.4 Warnings

Your code must compile **without error or warning** with `g++ -std=c++11 -Wall` on `ms8??` machines. However, we advise you to check your code also with `g++ -std=c++11 -Wall -Wextra`.

3.5 Evaluation

Your code will be evaluated based on all the above criteria. Failure to comply with any of the points mentioned in **bold face** can (and most often will) result in a mark of zero!

4 Submission

Projects must be submitted through the submission platform before **Friday, April 11th, 23:59 CET**.

You will submit a `s<ID>.tar.xz` archive of a `s<ID>` folder containing `othello.{cpp,h}`, where `s<ID>` is your ULiège student ID.

The submission platform will do basic checks on your submission, and you can submit multiple times, so check your submission early!