```
//
//  Connect4Evaulator.cpp
//  FreeRangeInterviewTest
//
//  Created by Stewart Bracken on 2/6/14.
//  Copyright (c) 2014 Stewart Bracken. All rights reserved.
//

#include <iostream>

//
//  Connect4.hpp
//  FreeRangeInterviewTest
//
//  Created by Stewart Bracken on 2/6/14.
//  Copyright (c) 2014 Stewart Bracken. All rights reserved.
//

#ifndef FreeRangeInterviewTest_Connect4Evaulator_hpp
#define FreeRangeInterviewTest_Connect4Evaulator_hpp

#include <vector>

typedef std::vector<char> conn4grid;

//Name of the game!
const int connect = 4;

const char red ='R',
           black = 'B',
           empty = '.';

class Connect4Evaulator {
    const int GRID_SIZE;

//****** PRIVATE METHODS *******//
private:
    int get_index (int x, int y, int _width);

    // Checks a tile against the next 3 using vx/vy as the direction.
    // Returns true if it's found a connect 4.
    bool has_connection4 (const conn4grid& connect4grid, char curr, int curr_x, int
curr_y, int vx, int vy, int width);

    //PRE: out_grid is empty
```

```
    void transpose (const conn4grid& grid, conn4grid& out_grid, int& width, int&
height);

    // Mirror flip rows
    void exchange_rows (conn4grid& grid, int width, int height);

    //Mirror flip columns
    void exchange_columns (conn4grid& grid, int width, int height);

    // Push all non-empty spaces downwards (increasing y)
    void apply_gravity (conn4grid& grid, int width, int height);


public:

    Connect4Evaulator(int grid_size = 42):GRID_SIZE(grid_size){}


//****** RETURN STATES *******//
    enum { RED_WIN, RED_LOSE, DRAW, UNFINISHED, NEITHER, LEFT, RIGHT, ERROR };


//****** PUBLIC METHODS *******//
    void print_grid (const conn4grid& grid, int width, int height);

    // Returns RED_WIN, RED_LOSE, DRAW, UNFINISHED, or ERROR
    int evaluate_conn4_state(const conn4grid& connect4grid, int width = 7, int
height = 6 );

    // Returns LEFT, RIGHT, NEITHER, or ERROR
    int evaluate_rolled_conn4_state (const conn4grid& original_grid, int width = 7,
int height = 6);


};


#endif


int Connect4Evaulator::get_index (int x, int y, int _width){
    return y * _width + x;
}

bool Connect4Evaulator::has_connection4 (const conn4grid& connect4grid,
                                           char curr, int curr_x, int curr_y,
                                           int vx, int vy, int width){
    for(int i=0; i < 3; ++i){
        curr_x += vx;
        curr_y += vy;
        if (curr != connect4grid[get_index(curr_x, curr_y, width)])
            return false;
    }
    return true;
}

void Connect4Evaluator::transpose (const conn4grid& grid, conn4grid& out_grid,
                                   int& width, int& height){
    //transpose by swapping rows with columns.
    for (int x = 0; x < width; ++x){
        for (int y = 0; y < height; ++y){
            out_grid.push_back (grid[get_index (x,y,width)] );
        }
    }
    std::swap (width, height);
}

void Connect4Evaluator::exchange_rows (conn4grid& grid, int width, int height){
    int half = height/2;
    for(int i=0; i< half; ++i){
        int row1 = i * width,
        row2 = (height - i - 1) * width;
        for (int x = 0; x < width; ++x, ++row1, ++row2){
            std::swap( grid[row1], grid[row2] );
        }
    }
}

void Connect4Evaluator::exchange_columns (conn4grid& grid, int width,
                                          int height){
    int half = width/2;
    for(int i=0; i< half; ++i){
        int col1 = i,
        col2 = width-1-i;
        for (int x = 0; x < width; ++x, col1+=width, col2+=width){
            std::swap( grid[col1], grid[col2] );
        }
    }
}
```

```
void Connect4Evaulator::apply_gravity (conn4grid& grid, int width, int height){
    const int NONE = -1;
    int last_empty_idx = NONE;
    int i;
    for (int x = 0; x < width; ++x){
        last_empty_idx = NONE;
        for (int y = height-1; y >= 0; --y){
            i = get_index (x, y, width);
            if (grid[i] == empty){
                if (last_empty_idx == NONE)
                    last_empty_idx = i;
            }else if (last_empty_idx  > NONE ) {
                //we've found an empty before and have a red or blue here
                std::swap( grid[i], grid[last_empty_idx] );
                last_empty_idx -= width; //up a row
            }
        }
    }
}


void Connect4Evaulator::print_grid (const conn4grid& grid, int width,
                                    int height){
    for(int y=0; y<height; ++y){
        for (int x=0; x<width; ++x){
            std::cout << grid[y*width+x] << " ";
        }
        std::cout<<std::endl;
    }
}

int Connect4Evaulator::evaluate_conn4_state(const conn4grid& connect4grid,
                                            int width, int height ){
    if (connect4grid.size() != GRID_SIZE)
        return ERROR;

    const int width_check_max = width - (width-connect)+1, //5 on standard grid
       height_check_max = height - (height - connect) + 1; // 4   "  "         "

    int result = DRAW;

    // Loop variables
    char curr; int i; bool has_connected_four;
    for(int y=0; y < height; ++y){
        for( int x = 0; x < width; ++x){
            i = get_index(x, y, width);
            curr = connect4grid[i];

            if ( curr == empty ){
                result = UNFINISHED;
                continue;
            }

            has_connected_four = false;

            if ( x < width_check_max ){
                //check across horz for a connect 4
                has_connected_four = has_connection4(connect4grid, curr,
                                                     x, y, 1, 0, width);
            }
            if (!has_connected_four)
                if ( y < height_check_max ){
                    //check down vertically for a connect 4
                    has_connected_four = has_connection4(connect4grid, curr,
                                                         x, y, 0, 1, width);
                }
            if (!has_connected_four)
                if ( x < width_check_max && y < height_check_max ){
                    //check diagonally down & to right for connect 4
                    has_connected_four = has_connection4(connect4grid, curr,
                                                         x, y, 1, 1, width);
                }
            if (!has_connected_four)
                if( y < height_check_max && x >= connect-1 ){
                    //check diagonally down & to left to c4.
                    has_connected_four = has_connection4(connect4grid, curr,
                                                         x, y, -1, 1, width);
                }

            if ( has_connected_four ){
                if ( curr == red )
                    return RED_WIN;
                else
                    return RED_LOSE;
            }
        }
    }

    return result;
}
```

```
int Connect4Evaulator::evaluate_rolled_conn4_state (
                                    const conn4grid& original_grid,
                                    int width, int height){
    if (original_grid.size() != GRID_SIZE)
        return ERROR;
    // Rotate 90 left == counter clockwise
    // Rotate 90 right == clockwise

    // To rotate first we transpose, by exhanging rows with columns.
    conn4grid trans;
    transpose (original_grid, trans, width, height);

    // Mirror flip rows at middle to rotate left.
    conn4grid left ( trans ); // Copy transposed so we can modify in place.
    exchange_rows(left, width, height);

    apply_gravity (left, width, height);
    int left_result = evaluate_conn4_state(left, width, height);
    if ( left_result == RED_WIN || left_result == RED_LOSE ){
        return LEFT;
    }

    // Mirror flip columns to rotate right.
    conn4grid right;
    std::swap(right, trans); // Don't need trans anymore so just swap it.
    exchange_columns(right, width, height);

    apply_gravity (right, width, height);
    int right_result = evaluate_conn4_state(right, width, height);
    if ( right_result == RED_WIN || right_result == RED_LOSE ){
        return RIGHT;
    }

    return NEITHER;
}
```