Assignment 2

Due date: April 26, 2015 (11:59 pm UTC+08:00)

Instructions and requirements – You are expected to submit two separate files, one presenting the high-level explanations of your algorithms, and a second including the code with in-line comments. You will be graded according to:

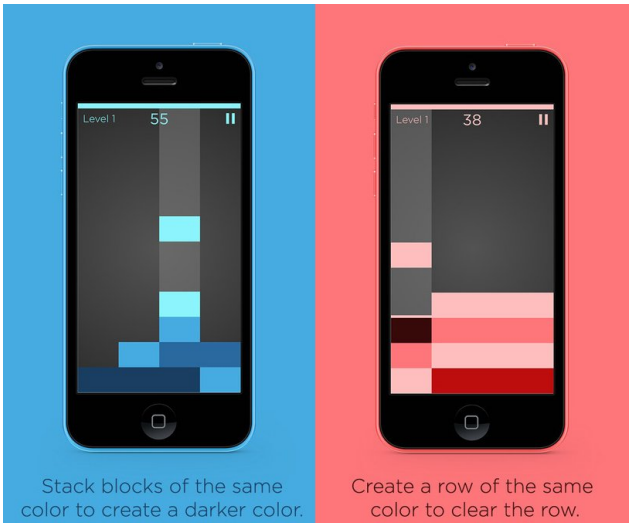
a. whether you implemented the algorithms **correctly**,

b. whether you have provided sufficient **comments** in your code, and

c. your **high-level** explanations of the algorithms.

Make sure your code compiles before considering submitting your report. Please ensure you have tested your code thoroughly, have documented all code you have written and have provided the proper explanations motivating both programs and the algorithms they use.

Shades is a game introduced in September 2014 for iPhone. The basic concept of Shades is similar to Tetris: The game consists in falling bricks that need to be matched together to avoid a pileup of bricks from reaching the top of the screen. The difference with Tetris is how bricks are matched against each other. In Shades, matching bricks is based on the shade of their color rather than the physical shape. Indeed, all bricks have the same shape and the same base color but vary in the shade of their color. There is a total of five (5) different shades.



Shades is played on a vertical board consisting of a grid of square spaces 11 rows by 8 columns. Each block consists of 2 side-by-side square spaces. The goal is to match two piled-up blocks with the same color shade, which merges the two blocks into one block of a darker color. Two blocks of the darkest shade will remain stacked unmerged. Playing blocks fall one after the other, a new

block appearing from the top of the board until they either reach the bottom of the board or the top of a previous block. If the two blocks have the same color shade they will be merged into a single block of a darker color unless the shade of their initial color is the darkest existing. After merge, the resulting block occupies the squares of the below block. If blocks cannot be merged, the falling block will sit on top of the below block or on the bottom of the board and will fill in the squares of the board that it occupies, The next block will then begin falling.

Using left and right arrow keys on the keyboard, the player can move a block left or right as it is falling. The player can thus control which column the block falls into by sliding the falling block right or left. Whenever an entire row of the board is filled with the same shade color, that row is removed and all the rows above it are moved down. Using the down arrow key drops the block as far as it will go without delay, so you don't have to wait for it to finish moving down gradually.

The objective is to keep playing as long as possible, which is accomplished by filling in rows with the same shade color in order to remove them. When there is no longer room on the board to drop another block, the game ends.

You may consider designing at least three (3) classes. A first class called Shades is the main class that will load a new board. It extends JFrame to create the board layout and contains all the main methods to run a new game. The Block class describes the color, the coordinates and other instance variables for each block on the board. It includes methods such as the one needed to determine the random color of the next falling block. The Board class extends JPanel and implements ActionListener. This class contains the logic of the game. The Board class runs a timer to repaint the board according to the failing speed of blocks and is in charge of tracking the key events form the user, clearing a full line of matching shade blocks, etc.