M256 Computer Programming
Major Project #5: *Frogger*

Setup: From the O: drive, copy the folder "FroggerLASTNAME" into your Canvas \ Projects folder. (Then, change the folder name to include your actual last name.)

In this project, you will complete a series of tasks to create a basic version of the *Frogger* game! The following four tasks will walk you through the barebones functionality of this game:

Task #1 (6 points): Create a log obstacle that goes from right to left (wrap)

Previously, we wrote code that displayed a rectangle on the screen and had it travel horizontally across the screen, wrapping around to the other side (*Image Display & Movement Exercise*). We also wrote code to display an image file on the screen (*Rectangle Class*).

For the current task, you will combine these behaviors to create a log object that travels from right to left. After it finishes scrolling off of the left edge, it should wrap around and reemerge from the right edge. The skeleton of the class (`logClass`) has been written for you. Write code to complete this class, and then instantiate one log object (place it somewhere near the top of the canvas) to test your code. For the log picture, download a picture of your choice from the internet (make sure to change the name of the image file to something short and convenient).

Task #2 (7 points): Create a car obstacle that bounces back and forth

For the current task, you will be writing a class called `carClass`. This class is similar to `logClass`, but instead of wrapping around the edge, it bounces back and forth. After writing this class, instantiate one car object to test your code (place it somewhere near the middle of the canvas).

1. A skeleton of this class has not been provided for you, so you will need to write it from scratch. Notice that your `logClass` code is a good starting point, since `carClass` is virtually identical to `logClass`, except that it bounces instead of wrapping.
2. For the car picture, download a picture of your choice from the internet. Make sure to change the name of the image file to something short and convenient.
3. Notice that if you use the same overhead car picture for both directions of travel, then the car can look like it's moving backwards some of the time. To address this issue, change the code in the `draw()` method of `carClass` so that instead of displaying the same picture all of the time, we display whichever picture is appropriate to the current direction of travel.

Task #3 (7 points): Create a frog character whose movement you can control

You will create a frog character whose movement you can control. The skeleton of the class (`frogClass`) has been written for you. Write code to complete this class, and then instantiate one frog object (near the bottom of the canvas) to test your code. Some notes:

1. Write code in the `gameUpdate()` function so that the frog moves left/right/up/down according to the arrow keys on your keyboard. Note that the frog should "hop" for each keypress, instead of moving smoothly.
2. Detect clean keypresses. In other words, you should *not* be able to just hold down the "Up" button and have the frog move up very quickly.
3. At this point, we are not doing collision detection, so you should be able to move the frog "through" the log or the car obstacles.

Task #4 (8 points): Gameplay behavior with collision detection

Implement the rest of the game play logic for playing Frogger:

1. If the frog object intersects with either the log object or the car object, the player loses and the game is over. Display a message on the screen notifying the player that they lost, and give the player the option of pressing a button to start a new game. Your implementation should use the `gameState` variable appropriately.
2. If the frog reaches the top of the screen without intersecting any other rectangles, then the player wins and the game is over. Display a message on the screen notifying the player that they won, and give the player the option of pressing a button to start a new game. Your implementation should use the `gameState` variable appropriately.

Possible Enhancements (4 points):

1. Keep track of the number of games that the player has won/lost and display this information.
2. Change the picture of the frog after each jump so that it looks appropriate to the direction of movement. For example, right after moving right, the frog is facing to the right; right after moving down, the frog is facing down; etc.
3. The player starts with 3 lives. If the player loses all 3 lives without making it to the top of the screen, the player loses; otherwise, the player wins.
4. Each time that the player wins, the game becomes more difficult. (Example: frog moves slower, log/car moves faster, log/car becomes larger, etc.)

Overall Code Quality (3 points): Variable / function names, indenting, commenting, etc.