Pong Game in C++ Worksheet

The skeleton code given will form a fully-fledged pong game in the command line when finished (beware, it's a bit flickery)

This project uses C++, a language very similar to C, but with some extensions and a few syntax changes

Some code is left in the skeleton to help you work out how to write the code

Concepts

- We'll be using classes to define things as objects
- Objects are useful as they can contain information about themselves, e.g. co-ordinates, size etc.
- Enums are datatypes that can only take certain values
- We have defined an enum `eDir` at the top of the project, which defines the 7 directions a ball object can have

Ball

- The `cBall` class defines the ball object in the game
- The ball will have old and new co-ordinates, as well as a direction
- Fill in the skeleton code for the `cBall` class, using the clues to help

N.B. The movement axis start from the top-left, so if you want to move an object downwards, you have to increase its y co-ordinate

Paddle

- The `cPaddle` class defines the paddle object in the game
- The paddle will have old and new co-ordinates
- It's worth bearing in mind in calculations (in this and future functions) that the paddle will have a width of 4 when drawn
- The paddle can only move up and down so its movement is a little simpler than the ball

Game Manager

- The game manager is the main class that is used to play the game
- It keeps track of positions and scores, and has pointers to the ball and each player

- The constructor and destructor (to erase the game from memory once it's been played) has been given
- The `draw()` function has also been given - it basically runs through each cell on the game board and checks if it needs to draw a wall, ball, player or blank space every time. It also prints out the players' scores
- The `input()` function takes input from the user's keyboard and does the appropriate action
- Note that it's important to see if a paddle has space to move before changing its position
- The `logic()` function deals with collisions (a ball hitting a paddle or a wall)

Finally, the `run()` function runs the game loop, continually drawing, taking input and checking for collisions, until the user quits

Main function

The main function will start when the program is run, and basically creates a `GameManager` object, and then invokes the run method to get the game started