**CS 210**

**Lab 13: Connect Four**

**Lab Date:** Wednesday, April 24

**Due Date**: Sunday, May 5, 11:59pm

**Submission**

* This is an **individual** lab.
* Submit all files in a single zip file to the Lab 13 assignment folder on Kodiak.

**Goal:** Implement Connect Four using the Model-View-Controller architecture.

**Instructions**

[Connect Four](https://en.wikipedia.org/wiki/Connect_Four) is a game similar to Tic-Tac-Toe. Two players alternate dropping tokens into a grid (7 columns x 6 rows), where tokens fall as far down into the grid as gravity allows. If there is no space available in a particular column, then a player cannot place a token there. The game ends either (a) when a player places four tokens in a row horizontally, vertically, or diagonally, or (b) when the entire grid is filled so that no player can make a move.

Use the Model-View-Controller software architecture to implement a version of the Connect Four game where players switch off using the same keyboard/mouse. Your submission should follow good design principles, with abstractions for the Model, Controller, and Observers (at minimum).

The GUI must show the following information at all times:

* The current state of the game board
* Whose turn it is
* In the event that the game ends, it must show which player won, or that the game was a draw.

Additionally, the GUI must allow users to perform the following actions:

* Pick which column to place a token in, while also disallowing placement of tokens in columns that are full
* Reset the game to an empty game board.
* Quit the game and exit the program.

In the event that the game ends, the GUI must notify the players which player won or that the game was a draw.

Note: You may represent tokens in any way you wish. You may use any combination of colors, shapes, and/or letters (Xs and Os) to show the board.

**Getting Started**

* What information does the model have to keep track of?
* What will the View and/or Controller want to be notified about?
* How will users interact with the board?
* How should user interaction affect the model? What does the controller need to know to pass on information appropriately?
* Start with either the model or the view.
  + If you start with the model, work out the game logic without considering the interaction beyond the fact that a move has to come from somewhere.
  + If you start with the view, figure out what the buttons (or other interaction components) are, but do not implement the event listeners. Get a dummy GUI going that looks nice but does nothing.

**What to Submit:**

* A zip file containing all source files.