ECE 421 Assignment 4 Part 1

1. On a computer it is possible to have an opponent for a single player. It can also check moves, only allowing valid ones, and check for win conditions. It can automatically keep track of the players' scores. It can keep track of the entire game state, so the players need not remember things like whose turn it is.

2. A computerized opponent is a simulated player that can make moves in the game without any input, based on the current state of the game. Typically its objectives will be the same as any player, to win the game. Computerized players should have a reasonable difficulty level, maybe even adjustable. For connect four it is important to have an opponent, and the option for either computerized or live players.

3. The colours of the play pieces and game board and text. The shapes of the pieces and board. The fonts of the text. Placement of messages. Background images. Animation of gameplay. Windows could be resizeable, with scrollbars added when content falls outside of the window.

4. One of the advantages would be that the GUI components could be ported to other systems using other programming languages. A possible disadvantage is the added complexity of another system. This kind of tool could be used early on in the development process, so the implementation could be driven by the completed interface.

5. In a GUI system, errors and messages can be displayed to the user via dialog boxes and other UI components as opposed to printing text to standard output.

6. Yes, we do. It is needed to write contracts for the application, and these will be implemented using a command line interface, rather than the GUI.

7. They both have the same game board, and they have similar pieces. The TOOT and OTTO pieces just have letters on them. Both games have a pattern of four pieces as a win condition. The turn system is the same for both games, alternating each move. The player, game board, and pieces can be modeled as objects which can be used for both game types. These models can then be reused in other configurations.

8. MVC is a framework for dividing code into models: things which contain data, views: things which display the data, and controllers: things which manipulate the data. We will use models to represent the various game components, views to display them using GTK, and controllers to update the game state.

9. The strategy pattern will be used to check for the win condition in a game. It will be able to check for the different types of four in a row patterns. Our views will be observers, which observe our models. The controllers will be called from interaction with the view, then the controller will update the models, and this change will then be reflected in the view. See 8 for MVC definition.

10. We will be using the Gtk namespace to access all of its class variables. Other than that we will not be making or using any other namespaces.

11. Checking for wins will require an iterator. An iterator will also likely be used to update all the views when a model is changed.

12. Standard error will apply to this problem, and some of its subclasses like IndexError for if a player makes an illegal move in the game board. An exception pattern that could apply is AvoidExceptionsWheneverPossible.