Due date: **November 17th, 10:30am**

For this project, you will implement the game 326tac-toe using Python and its Tkinter GUI library.

Overview:

- **Your program must run on Python 2.5.2 on the ECF workstations.**
- Your program must a have an attractive and clear GUI, and your mark will be partly based on the quality of the GUI.
- You must provide a user manual intended for users who are computer-literate but don't know anything about programming. Your mark will be partly based on the quality of writing in the manual.
- You must work in groups of size at most three. Each group should send me an email with a list of group members. The deadline for this email is **October 20th**.

The game:

You will implement the game 326tac-toe, a generalized version of tic-tac-toe.

Recall that tic-tac-toe is played on a 3x3 board, and the first player to get 3 in a row – horizontally, vertically, or diagonally – wins. If the board fills up without any player getting 3 in a row, the game ends in a draw.

In 326tac-toe, the board is 8x8, and the first player to get 4 in a row – horizontally, vertically, or diagonally – wins. As in tic-tac-toe, if the board fills up without any player getting 4 in a row, the game ends in a draw.

The GUI:

Implement the GUI for your game using Tkinter. All user interaction must be through the GUI. The GUI should be easy to understand even for someone who hasn't read the manual.

The program should ask for the names of both players. The program should always clearly indicate whose turn it is. When the game ends, the program should display a message congratulating the winner (or announcing a draw), and allow the user to either quit or start a new game.

Use the Tkinter Canvas widget to draw the game board. To play in a particular position, a player should just have to click inside that position on the board. Use circles (of different colours) to indicate the moves of each player (instead of using Xs and Os).

To ensure that your game is playable even on smaller displays, your program's windows should be no larger than 650x650 pixels.

Implementing an additional feature:

You must implement ***one*** of the following features.

- Give the user a choice of board sizes and the number in a row needed to win. The set of board sizes the user can choose from must include 5x5, 6x6, 7x7, 8x8, 9x9, and 10x10. The choices for the number in a row needed to win must include 3, 4, and 5. Note that if your code for the standard game is well-designed, this feature will be relatively easy to implement.
- Allow the user to save a game in progress (to a file), and to load saved games (from a file) and continue them. This must preserve all information associated with the game in progress, including the players' names.
- A feature of your choice. The level of difficulty should be about the same as the two features suggested above. You must get your feature **approved** well before the due date. Send me an email with a clear description of your proposed feature.

Style and design:

Follow the usual programming style rules. This includes making good decisions about object-oriented design. You'll find that good design (including the appropriate use of exceptions) will make this project much more manageable.

Use a doc string to comment each module, class, method, and function. Use additional comments where necessary. Choose meaningful names for variables, functions, methods, and classes.

User manual:

Your manual should be a brief document (no longer than a page – most users won't read anything longer than that) telling users how to run your program. See the "Submission" section for acceptable file formats and file names. Remember your target audience: users who know how to use a computer, but don't know anything about programming. This means that you shouldn't be giving any technical details about your implementation.

Marking scheme:

The marks will be allocated as follows:

- 30: Successful execution (of required features)
- 10: Additional feature
- 15: Style and design of source code
- 10: GUI design and user experience
- 5: Helpfulness and writing quality of user manual

Tkinter:

Learning how to use Tkinter is part of this project. Tkinter will not be covered in detail in class, though (time permitting) there might be a *brief* overview. To start learning about Tkinter, look at the following:

http://www.ferg.org/thinking_in_tkinter (Thinking in Tkinter)

http://www.pythonware.com/library/tkinter/introduction (An Introduction to Tkinter)

http://infohost.nmt.edu/tcc/help/pubs/tkinter (Tkinter reference: a GUI for Python)

http://tkinter.unpythonic.net/wiki/Tkinter (Tkinter Wiki)

Advice:

This project may be somewhat large, but it's not meant to take over your lives. It makes a lot of sense to build something small that solves part of the problem, and then add to it as you work. Get a text-based (non-GUI) version of the game working first. Separately, develop the GUI (without any game logic). Then, combine the game logic and the GUI. Working on the game logic and the GUI separately will make debugging much easier.

Make sure you always have something to fall back on, so that you can submit a working program.

Academic offenses:

The standard rule is in effect: don't share your work with any other group.

Submission:

How to submit

Submit your work using the command `submitcsc326f`. Set the first argument – the "assignment#" – to **4**. Only **one group member** should electronically submit. If you submit files using different logins, send me an email stating which submission should be marked (otherwise, we'll have to arbitrarily choose one of the submissions to mark).

What to submit

- The Python code, in however may files you like. The main module must be in a file called `game.py` (that is, we should be able to run your program by running `game.py`).
- The user manual, in a single file called `manual.`*ext* where *ext* is either `txt` or `pdf`. This means that your manual must be either a plain text file or a PDF document.
- A `readme.txt` file documenting any bugs or missing features.
- Both `game.py` and the user manual must include the names and ECF logins of all group members.