**CSCE 155N Matlab Programming Project 1 – Fall 2013**

**Assigned: Thursday 9/19/2013**

**DUE DATE:**

**Tuesday 10/8/2013 at 11:59 PM**

**(hardcopy in class on Thursday)**

**Slick-Slack-Sloe**

**Problem Statement:**

The game of Slick-Slack-Sloe is a variant of Tic-Tac-Toe. The two-dimensional square board generally has 4 rows and 4 columns, but this can be set by the players. Just as in Tic-Tac-Toe, two players alternate placing their marks (X and O) on the board in an attempt to fill in an entire row, column, or diagonal with just their mark.

Unlike Tic-Tac-Toe, on each turn the player may choose to slide a row or column one position in either direction instead of placing a mark on the board. In fact, once the board is filled, this is the only possible way to move. Sliding results in all the pieces on that row (or column) shifting over one position, and the piece that drops off cycles back around to the far end from where it drops. The one constraint when choosing the row or column to slide is that it must contain at least one of that player’s marks.

The winner is the player whose marks are the first to complete a row, column, or diagonal. A tie is possible if both players complete a row, column, or diagonal simultaneously as a result of a sliding move.

You are to work in teams of two or three students to design prototypes of Slick-Slack-Sloe in Matlab. Larger teams may be allowed by permission, but there would be higher expectations.

**Collaboration:**

Work together as a class on any or all aspects of the research and design. Ideally take advantage of the talents of each member of the team, but recognize that each is responsible for the entire project! This means being prepared to answer questions on the code even if your allocated task was to write the report. It is essential to keep track of who did what and where any useful information was found. Record each time you helped someone else and each time someone helped you. Keeping a log is highly recommended. Note that Piazza keeps a record automatically!

**What and How to Submit:**

Read and have your program conform to the “Program Documentation Guidelines” which are online.

By the deadline hand in electronically the two files, sss.m (the Matlab script file for the game), and sss.doc (which contains summaries, documentation, and sample runs). Only one copy per group needs to be submitted – it does not matter who does the handing it. In the first classday after the deadline, hand in the hardcopy version, stapled together with the cover page in front.

Each team member should electronically on his/her own account submit his/her own analysis of the relative contributions of all the members toward the project. This is in addition to the acknowledgement section of the main report. Assuming allocation is fairly even, all will receive the same grade.

The Word document should contain the following, all carefully labeled:

* Cover page with name(s) and the account under which it is submitted, title, date submitted, etc.
* A discussion of the features you implemented in the project. Describe how they work and what Matlab options were used to program them. This should be at a fairly high level, not a line-by-line analysis of the code.
* An “instruction manual” that a non-programmer can use to run the game.
* An annotated cut and paste sample dialog of the running of the program. (Hint: Use the ‘diary’ command and/or capture screen shots.)
* A discussion of the testing that was performed. This should include testing of each component as it was being built, and testing of the final program ensuring that it works properly under a comprehensive range of conditions.
* An annotated cut and paste of a sample dialog, demonstrating how your program responds to extreme and faulty input. (This could be combined with the previous section.)
* Acknowledge all collaborations (both internal to the team and external), detailing what each person contributed individually, and what was done jointly. Indicate approximate percentages of the work contributed by each person in design, coding, testing, documentation, and report preparation.

**Grading Criteria:**

* Program functions as intended – 30%
* Program logic is well designed – 20%
* Documentation guidelines are followed – 20%
* Handin Documents formatted and arranged as specified – 10%
* Testing is comprehensive – 10%
* Quality of the user’s manual – 10%