**CSCE 155N Matlab Programming Project 1 – Summer 2017**

**Assigned: Monday 7/17/2017**

**DUE DATE:**

**Friday 7/28/2017 at 11:59 PM**

**(hardcopy in class Monday)**

**Mattetris**

**Problem Statement:**

Tetris has been a popular video game for decades. We will be implementing a simplified version which the computer simulates rather than having a human control. Considering the implementation is in Matlab, the name will be Mattetris.

The idea of Mattetris is that rectangular blocks will be fitted together into as tight an area as possible. Input will be the dimensions of the blocks. For each game, the player will indicate one of three possible sources for the dimensions: 1) interactive human input into a n x 2 matrix, 2) randomly generated by the computer, or 3) input data file to be loaded. Rules of play are as follows:

1. Make the screen 80 characters wide and as high as needed to display all the rectangles.
2. Rectangles are placed in the order provided in the input.
3. Rectangles may be placed oriented as given (preferred), or rotated 90 degrees clockwise.
4. Each rectangle should be placed with its top line as high on the display as possible.
5. Within that highest line, each rectangle should be placed as far to the left as possible.
6. After it is placed, a rectangle will not be moved.

Rules for the display of a rectangle are as follows:

1. The characters forming the rectangle are numbered sequentially row by row, column by column, (e.g. as a Matlab matrix on input of 4 3 it would be [1 2 3; 4 5 6; 7 8 9; 10 11 12]).
2. Only the 1’s digits are displayed, (e.g. for the above as [1 2 3; 4 5 6; 7 8 9; 0 1 2]).
3. If the rectangle is rotated clockwise 90 degrees, the display should likewise be rotated, (e.g. for the above as [0 7 4 1; 1 8 5 2; 2 9 6 3]).
4. Only the edge of the rectangle is displayed, (e.g. for the above do not display the 5 and 8). Interior positions show as spaces.

**Problem input:**

You may choose to accept input from the keyboard, either prompting for each rectangle, or prompting for an entire list of rectangles. Alternatively you may choose to accept input from a data file. Finally you may choose to specify some number of rectangles to be generated by the built-in random number generator. Perhaps you can think of other ways to get the data into the program. In general, each rectangle should be entered as height width pairs (e.g. 6 9 for six rows high and nine columns wide). If you upgrade this project to become project 4, you might use a GUI approach to data entry.

**Problem output:**

Similarly, you may design how the results are reported. We should at a minimum see the final arrangement of rectangles being displayed. If you upgrade this project to become project 4, you might use animation and a figure to show the rectangles. Again, discuss your choices in the report.

**Problem options:**

You are welcome to propose variations, such as implementing smarter algorithms for rectangle placement, allowing more complicated shapes, etc. Yes, discuss your choices in the report.

**Collaboration:**

Work together as a team on any or all aspects of the research and design. Ideally take advantage of the talents of each member of the team. Use your teams to finalize each of the multiple design options. It is essential to keep track of who did what and where any useful information was found. You need to keep track when you help someone and when you receive help from someone. This includes students from other teams and those outside the class, TAs and me (the instructor). Keeping a log is highly recommended.

**What and How to Submit:**

Read and have your program conform to the “Program Documentation Guidelines” which were provided previously.

As you make progress with the project, periodically handin (electronically) updates numbered as mine are on my site. How many updates really depends on what seems natural. It might be 5 or it might be 15. Anyway, we should see a progression of “working” programs handed in as ***mattetris1.m***, ***mattetris2.m***, etc. Choose just one member of the team to do the submitting, but make sure all names are listed in all submitted files.

By the deadline hand in electronically the two files, mattetris.m (the Matlab function file for the simulation), and mattetris.doc or .pdf (which contains summaries, documentation, and sample runs). In the next Monday class, hand in hardcopy versions, stapled together with the cover page in front.

Each team member should submit his/her own analysis of the relative contributions of all the members toward the project. This should be submitted electronically using each member’s handin account. 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 set up and run the simulation.
* An annotated cut and paste sample dialog sampling of the running of the simulation. (Hint: Use the ‘diary’ command or cut & paste as appropriate.)
* 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:**

* Properly running features – 30%
* Program logic is well designed – 20%
* Progress versions – 20%
* Documentation guidelines are followed – 10%
* Handin Documents formatted and arranged as specified – 10%
* Testing is comprehensive – 10%