CS-3398 Fall 2018, Team Project 1: “A Game of Checkers”

# Problem Description:

Your team will need to implement a “Game of Checkers”. Requirements not explicitly laid out in this document must be captured in the Software Requirements Specification (SRS). Design and implementation of a solution to these requirements must be laid out in the Software Design Specification and Code and Unit Test. A Test Plan and Test Results, with a list of recorded defects, must also be written and presented. You MAY use libraries/APIs, but you MAY NOT copy code. All documentation and code must be pushed to GitHub by everyone as they complete work (exceptions need to be run by me). Provide any run instructions to me via .md file in GitHub.

# High level minimum requirements:

The delivered Code **must** compile (if a compiled language is used) and execute. The goal for this project is to understand the process, not to deliver the highest quality, most robust, or most full featured product. However, a minimum of the following four requirements must be implemented in each waterfall phase for full credit (following the American Checkers rules found here https://simple.wikipedia.org/wiki/Checkers#Rules).

1. The game shall provide a user interface (ASCII text is fine, sprites, html images, etc.).
2. The game shall leverage at least ONE form of AI implementation to play against the user.
3. The game shall begin from the normal starting position and must indicate a win/loss state when it has determined a winner.
4. The game shall enforce piece position and movements.
5. The game shall allow “kings”.

# Grade Breakdown:

* SRS – 20 points
* SDD – 20 points
* Code/Unit Test – 20 points
* Test Plan/Test Results – 20 points
* Individual Analysis – 20 points
* 360-degree evaluation - +/- 10 points

Grading for each section will depend on its completeness and correctness. In each phase, meeting the **high level minimum requirements** with perfect implementation will get you 16 points for that phase***. Each additional requirement added and met will earn your team 1 additional point \*per section\*.***

**Example A:**

*Team A implements a checkers game that satisfies all the high level minimum requirements but has added no other requirements or features. Implementation and documentation is solid. Each member of the team will receive:*

* *16 points for the SRS*
* *16 points for the SDD*
* *16 points for the Code and Unit Test*
* *16 points for the Test Plan and Test Results*

*So, if the Individual Analysis and 360-Degree Evaluation are perfect, the most points a team member can get is 94.*

**Example B:**

*Team B implements a checkers game that satisfies all the high level minimum requirements and has decided to add a configurable second AI algorithm (new requirement 1). They have also decided to add a two-player mode (new requirement 2), and an optional Continental draughts (https://simple.wikipedia.org/wiki/Checkers#Continental\_draughts) mode (new requirement 3). The team was able to capture all of this in the SRS and SDD, but they could not quite implement the rules system in code and test. That teams break down would be:*

* *19 points for the SRS*
* *19 points for the SDD*
* *16 points for the Code and Unit Test*
* *16 points for the Test Plan and Test Results*

*So, if Individual Analysis and 360-Degree Evaluation are perfect, the most points a team member can get is 100.*

# SRS:

You must write an SRS capturing the functional requirements from the high level minimum requirements and any optional requirements your team has decided to add. For this project, I will be your customer. If you would like to add a requirement, you need to run the requirement by me for approval. It is very unlikely that I will deny any requirement, but I may warn against them if I think the implementation will be too much work or that the feature does not make sense or is inappropriate.

Expect to write between 5 and 10 pages. If you use any outside references, **PLEASE CITE THEM APPROPRIATELY!** Any features designed, implemented, or tested that are not in the SRS will not count. This phase is the cornerstone of the product.

# SDD:

You must write an SDD detailing the high to mid-level design of your system. Use UML when applicable and try to capture enough detail to show that you have analyzed how the system should be built. Good rule of thumb, if you can give me this document and I can code it without any questions, you have done well.

Try to use design patterns during this phase.

This document will likely be longer. It could be anywhere from 5 to 30 pages, but it will mostly consist of diagrams and class descriptions. You should not need to reference anything here, but if you do, please CITE THEM!

# Code/Unit Tests:

You must implement the design you wrote. Make sure you implement this based on the SDD, not on the SRS! If you have find a significant problem with the requirements or design during this phase, make sure you go back and iterate on the SRS/SDD! Welcome to waterfall! If you do not do this, the SRS will not match what you coded, and you will not get credit for that feature.

The code must compile and execute.

I also want some unit tests implemented. Given time constraints, I will not ask you to do more than a few. Focus on testing your algorithm(s) and not the objects for the pieces, board, etc.

# Test Plan/Test Results:

A high-level Test Plan must be written and the test run. Results must be presented. Please capture at least two defects. I do not expect you to fix the defects unless you think they are critical (Priority 1). Defects like this would make the game unplayable, and your score for the Code/Unit Tests phase would suffer.

# Team Results Analysis:

I would like a team driven analysis (what we call post-mortem) of the project. Two or three pages are fine. Please consider and present what challenges your team faced during this project. Also, describe any consensuses or disagreements your team had. For example, did you debate adding a feature? Did you have disagreements between testing and requirements? This is common, and it is good to be aware of them.

# Individual Results Analysis:

Finally, I would like a short, one-page narrative on your thoughts of the process and project. What parts did you enjoy? What was your biggest complaint? What would you have done differently?