Effort Report

Work done: All work was done my Kent Sommer (me).
The following was done in python using a pre-existing base code set.

1. Implemented MiniMax search
2. Implemented alternative MiniMax with Alpha Beta pruning (buggy)
3. Implemented 3 different Heuristic/Evaluation functions.
4. Set testing with different Heuristics to compare quality/usefulness and see weaknesses/problems.

Tools used:
All code except for base code to handle player input and board drawing was written by myself (Kent Sommer)

1. http://inventwithpython.com/reversi.py (base code used to handle board drawing and player input)
2. https://github.com/kentsommer/Reversi-AI (modified code base including all AI implementations and alternative heuristics)

Sales Pitch:
At face value, midway through the semester, it was obvious to me that minimax would be the best choice to implement an AI to play reversi. What I did not realize at that point in time, however, was how important the heuristic function would be as well as how many different heuristics you could potentially use. The problem suddenly became much harder than I had originally thought it would be (including much more math…). It was no longer a matter of simply implementing minimax with Alpha Beta pruning, but instead became a search for a great heuristic and the challenge of implementing it. Because of how important the heuristic function is, the problem is not as simple as it appears. Since Reversi is zero-sum, one could argue that the game can be solved (perfect play), however, it isn't that simple. After my work, I believe it CAN be solved, however, the depth you would have to search to would make it virtually impossible to implement (with current hardware). Most of my time (30 hours) was spent working on and researching the evaluation heuristic functions. A small amount of time (5-6 hours) was spent learning to play the game at a reasonable level. A decent amount of time (14 hours) was spent writing and debugging my code for Minimax as well as the Heuristic functions. The last chunk of my time was spent testing heuristics/my AI (4 hours).

I honestly was not expecting this project to take nearly as many hours as it did. I ended up restarting once after using a different code base which was severely bugged and I did not realize it till I had already written a substantial amount of code for it (C#). I ended up switching to a python code base since it was shorter and actually worked properly. I'm not sure how best to go about showing effort in research other than to say to look at my topic paper for the heuristics. I spent more time than was probably needed to do testing but that was because the code base I used only allowed for player vs AI so I ran two instances in order to artificially get the AI to play itself (in hindsight I could probably have just modified it to play against itself which would have saved me an hour or two, but doing it slowly by hand also allowed for more granularity in detail about piece gain and loss).

Lesson learned:

I think the most important thing I will take away from this, is that search doesn't really make a difference for how good the AI is for this game. Minimax search is great, but without a heuristic or board evaluation function it is almost no better than random. Most of my time was spent tweaking and developing my evaluation function and that should be apparent looking at my talk through heuristics in the topic paper.