Ben Whitley
September 9, 2015
CSCE 420
Peg Game Assignment

Unfortunately, the program runs out of memory before completion. The objects used to create the tree are fairly large. There are multiple objects in this program and many helper functions. There is a PegSpace object which defines a space in a PegBoard. The PegSpace contains a space number, a Boolean to determine if it's occupied, and a list of valid steps. There is a PegBoard object which contains a list of 15 PegSpace objects (or 21 if you decide to add more). This is where moves are performed. Finally, there is a PegGameTree which contains nodes for PegBoard states. The head node is the initial board state, and the children nodes are the PegBoard states after a valid move. Unfortunately, due to time constraints, there are multiple PegGame states in the tree because I didn't check to see if the state already exists before adding it. I believe this is where the memory issue arises. The PegGameTree has a recursive function Populate which runs until all possible moves are completed, or, in my case, the memory because too full to run any longer.

The program runs in Visual Studio 2012. Double-click on AI_PegGame.sln in the AI_PegGame folder to open the solution in VS2012. Compile and run the program.

I didn't use any resources to complete this project.

Aggie Code of Honor:

An Aggie does not lie, cheat or steal or tolerate those who do.