Project Part 1 Summary

Please submit 1 per team via Moodle (please have only 1 person on the team submit). It is recommended to have this on Github too, in case someone forgets to submit it on Moodle.

TEAM NAME: C3W

Algorithm + Problem	**TOOK OWNERSHIP name**	**userID**
Backtracking DFS + Cross-Math	Benjamen Weber	webe0491
A* + Sliding Puzzle	Nathan Comer	comer028
Genetic Algorithm + n Queens	Lee Williams	will4379
Simulated Annealing + n Queens		
Bidirectional Search and Iterative Deepening Search + Tower of Hanoi	Noah Wong	wongx565
A* + Path Planning (Striving for an A)	Nathan Comer	comer028
Backtracking DFS + Cryptarithmetic (Striving for an A)	Benjamen Weber	webe0491
Applying Genetic Algorithms to Towers of Hanoi (Striving for an A)	Lee Williams	will4379

Backtracking DFS **Benjamen Weber -** + Cryptarithmetic (Striving for an A)

Problem Implementation Complete: **YES**
Problem Implementation Correct: **YES**
Algorithm Implementation Complete: **YES**
Algorithm Implementation Correct: **YES**

KNOWN BUGS or INCOMPLETE PARTS: Cryptarithmetic was only partially completed.

COMMENTS: Modified node class to add `expandGenerator()` and `makeChildGenerator()` so that generators could be used in Backtracking DFS algorithm. These generators make sure only one child for a node is expanded at a time.

A* (Nathan Comer)

Problem Implementation Complete: **YES** NO
Problem Implementation Correct: **YES** NO MOSTLY

Algorithm Implementation Complete:	**YES**	NO	
Algorithm Implementation Correct:	**YES**	NO	MOSTLY

If this was done collaboratively, please estimate the percentage of each team member's contribution to this algorithm+problem: No collaboration

KNOWN BUGS or INCOMPLETE PARTS: None

COMMENTS:
Should work correctly for all sliding puzzles and sizes as long as they are square. Tends to take a little while for larger puzzles. I'll work on making it more efficient for part 2.

GA or SA (Lee Williams)

Problem Implementation Complete:	**YES**	NO	
Problem Implementation Correct:	**YES**	NO	MOSTLY
Algorithm Implementation Complete:	**YES**	NO	
Algorithm Implementation Correct:	**YES**	NO	MOSTLY

If this was done collaboratively, please estimate the percentage of each team member's contribution to this algorithm+problem: no collaboration

KNOWN BUGS or INCOMPLETE PARTS: the fitness heuristic used by GATower rewards local maxima too highly, and prevents further progress. Will revise this if I get a chance or leave this for later research. GATower needs more extensive commenting.

COMMENTS: On the striving part, I worked on adapting a Genetic Algorithm for variable length chromosomes for problems like Towers of Hanoi. I also coded my own Towers of Hanoi problem class to add methods needed for a genetic algorithm, which I named GATower and appears in GATower.py
Class GAQueens solves nQueens, class GATower solves Towers and incorporates adaptive operator fitness (see Davis: Handbook of Genetic Algorithms) to evolve the selection of operators as the run progresses by rewarding operators that make useful new members, and punishing those that make useless ones. Operators are then chosen by roulette wheel selection.

Bidirectional + IDDFS Noah Wong

Problem Implementation Complete:	**YES**	NO	
Problem Implementation Correct:	**YES**	NO	MOSTLY
Algorithm Implementation Complete:	YES	**NO**	
Algorithm Implementation Correct:	YES	NO	**MOSTLY**

If this was done collaboratively, please estimate the percentage of each team member's contribution to this algorithm+problem:

KNOWN BUGS or INCOMPLETE PARTS: IDDFS isn't complete, it stops working, Bidirectional works but it just returns the middle node not the path.

COMMENTS:

Didn't make solver files classes
Didn't use AIproblem as a parent class
Trouble using problem in a recursive function (IDDFS) tried to work-around by creating a new Towers everytime, probably caused issues with the function
Has a print function for troubleshooting issues

Path Planning using A* in MazePuzzle class (Nathan Comer) - Striving for an A problem implementation

Problem Implementation Complete:	**YES**	NO	
Problem Implementation Correct:	**YES**	NO	MOSTLY

Uses the same A* algorithm as my sliding puzzle problem.

If this was done collaboratively, please estimate the percentage of each team member's contribution to this algorithm+problem: No collaboration

KNOWN BUGS or INCOMPLETE PARTS: None

COMMENTS:
-Argument 'size' must be provided in the MazePuzzle constructor
-Does path planning on binary maps like those at: http://www.movingai.com/benchmarks/
-Three binary maps are provided in the problem directory as examples
-Binary maps from the website must have any additional data removed (everything except the actual puzzle/map)
-Upon running the algorithm on the puzzle the initial state will be printed. Once the optimal solution is found the state will be printed again with the planned path indicated on the puzzle

Binary map requirements for compatible:
-Binary maps must be square (height = width)
-Must be a text file consisting of a grid with only two types of characters:
    '.' character representing passable area/terrain (hallway, etc)
    '@' character representing impassable obstacles (wall, mountain, etc.)

Further Explanation:
-maps are traversed starting at [1,1] (the top left corner) to [size-2,size-2] (the bottom right corner)

FOR ANY ADDITIONAL WORK, please provide a similar format to that above.