To-do list:

\*\*Search "TODO" in the code files to find the areas that needs to be worked on\*\*

Cell.java

* calculateDistance() - calculates hValue using some heuristic (not sure which tho)

Main.java

* initialize() - generate random goal[] that is at least minGoalDistance away from start[]. Will probably have to use calculateDistance()

HeuristicSearch.java

* search() - this should be the main algorithm to find shortest path for all the types of searches

\*\*Done for now\*\*

UniformCost.java

A\_Search.java

AWeightSearch.java

Note:

* These classes initializes the PriorityQueue<Cell> queue in HeuristicSearch.Java with a Comparator on different Cell fields so the queue will order differently based on which subclass HeuristicSearch is. (See comments in code)
* The queue should be used in HeursiticSearch.java search()

Using HeursticSearch/UniformCostSearch/etc.. class example:

HeuristicSearch uniformCostSearch = new UniformCostSearch();

ArrayList<Cell> uniformCostSearch\_ShortestPath = uniformCostSearch.search();

HeuristicSearch aSearch = new A\_Search();

ArrayList<Cell> aSearch\_ShortestPath = aSearch.search();

\*\*GUI Notes \*\*\*

* Start has a red outline, goal has a blue outline