CS303 extra credit Assignment
https://github.com/Jordinaa/astart_rdj

Project Members: Jordan Taranto, Riley Connors, Dylan Frazier

● Graph Representation:

○ The campus network is represented by a weighted, undirected graph

○ Appears in the user interface as a two dimensional grid

○ The grid extends from (0, 0) to (x,y)

○ Each vertex is a location

■ Each location is distinguished by its own individual coordinates between (0, 0) and (x,y)

○ Each edge is the pathway between the individual locations

■ Weights represent the distance between the edges

Algorithm used: A*

○ Three parameters:

■ g: cost of moving to an adjacent node

■ h: heuristic value, estimated cost to goal from a given node

■ f: sum of g and h

○ The algorithm chooses the adjacent nodes with the smallest f until it reaches the goal

○ Implementation in project:

■ Struct: Node

● Contains int data fields x and y for the coordinate system

● Contains float data fields for f, g, and h parameters

● Contains a pointer to parent node

● Defines == comparison operator for nodes to check if two nodes have the same x and y

■ Struct: Compare

● Used to make and compare items in priority queue openList

● Compares two nodes' f data field

■ Struct: HashFunction

● Used to make unordered_set closedList

● Generates keys for the unordered set based on x and y coordinates

■ Float function: heuristic

● Calculates and returns the heuristic value from one node to another

● Used in the program to calculate estimated cost to goal from the current node

■ Bool function: findPath

Creates priority queue openList

○ Used to store nodes being processed

○ Ordered by nodes' f data field

● Creates unordered set closedList

○ Used to store nodes that have been fully processed

● Creates start and goal nodes for the user's chosen starting and ending point

● Pushes the start node to openList

● Continually processes nodes in openList and moves them to closedList, while adding the current node’s neighbors to openList

○ Processes nodes with smallest f first

○ Does not process nodes in closedList

○ Continues until goal is reached, then displays path and returns true

● If openList becomes empty (failed to find goal), returns false

How to use the application:

○ On startup:

■ User inputs dimensions of grid (Size of X and Y)

■ Graph with randomly generated weights is created from these dimensions

○ The user can select starting and ending points via command line

○ Three commands:

■ Choose grid size

● Allows the user to choose what size the grid will be

● Also generates the random weights for the graph

■ Choose starting point

● Allows the user to choose the starting point

■ Choose destination

● Allows the user to choose the destination (goal of pathfinding algorithm)

○ Risk mitigation

■ If user inputs a starting point or destination not within the graph, an error message is shown and the user is prompted to enter a valid coordinate

○ Output

■ Displays shortest path

■ Displays total distance required to traverse path