ME 459/5559 Exam # 2
Fall 2022
DUE Wednesday, November 16th, 2022 at 11:59 PM

Rules:

1. You **MUST** work individually.
2. Any suspected cheating will be immediately reported to the associate dean.
3. The exam is a take-home exam.
4. Additional written and/or oral justification for any answer may be requested by the instructor.

I have read, understood, and agree to abide by the exam rules.

Printed Name: _____

Signature: _____

1. **Conceptual Questions**

   (a) Explain in what scenarios would RRT be the best choice (of the RRT, A*, Dijkstra's options). 5%

   (b) If you have a single starting point and 20 stops on your delivery run (traveling delivery man). How many possible paths are there that visit each of the stops? Make sure to show your work. 5%

   (c) Describe how the genetic algorithm operates, and how it is applicable to the traveling salesman problem. 5%

2. **Trajectory Generating and Following Evader** 30%

   Your objective is to incorporate your Proportional Navigation pursuer turtlebot with an evader that generates and follows an optimal path from a start to the goal. **You must provide a plot of each bots' path along with the desired trajectory generated for the evader.** You are not required to include any obstacle avoidance into your pursuer path planning, but may find some small modifications are required to successfully chase (and catch) the evader prior to the evader reaching the goal location as well as make sure your evader does not collide with any obstacles. **In addition to the plot(s), provide the code used for the problem, and provide any comments/discussions necessary to interpret your results and describe any modifications you had to make in order to be successful.**

   Launch File: *two_turtlebot3_test.launch*
   World File: *umkc_test.world*
   Obstacle List: $(5,0),(5,1),(5,2),(5,3),(5,4),(0,5),(1,4),(2,3),(3,2),(3,3)$
   Evader Parameters: Name - *turtlebot1*, Start - $(2.0,1.0)$, Goal - $(7.0,2.0)$, Max Speed - $0.15m/s$

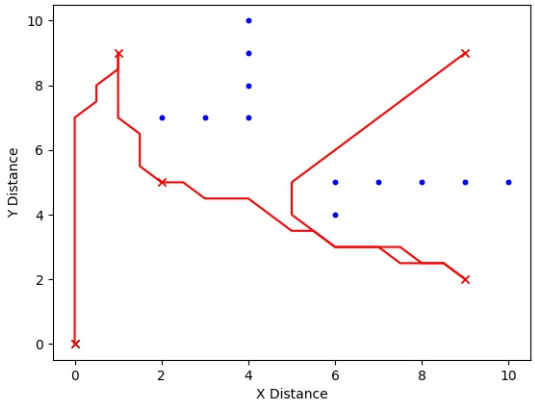
Pursuer Parameters: Name - *turtlebot2*, Start - $(0.0, 0.0)$, Maximum Speed - $0.2m/s$
Algorithms: Pursuer - *Proportional Navigation*, Evader - *Dijkstra's Algorithm*

3. **Traveling Salesman Problem with A*** 25%

Your objective is to use the *A* Algorithm* to navigate through the modified maze (obstacle list posted below) to deliver five packages. You must start at (0,0) and visit each of the delivery points. Provide a plot showing the obstacles, A* paths, and delivery locations. Provide the total travel distance of your solution. How does this compare with the other possible paths (how good is the solution, are there other paths that have much higher travel costs)? Turn in a copy of your working code.

Grid Info: Xmin = 0, Xmax = 15, Ymin = 0, Ymax = 15, Grid Size = 0.5, Robot Radius = 0.5
Delivery Points: (9,4), (4,4), (1,9), (9,7), (6,14)
Obstacle List: (2,2), (2,3), (2,4), (2,5), (0,5), (1,5), (2,5), (3,5), (4,5), (5,5), (8,2), (9,2), (10,2), (11,2), (12,2), (13,3), (8,4), (8,5), (8,6), (8,7), (8,8), (8,9), (8,7), (2,7), (3,7), (4,7), (5,7), (6,7), (7,6), (9,6), (10,6), (11,6), (12,6), (15,8), (2,9), (2,10, (2,11), (2,12), (2,13), (5,9), (5,10), (5,11), (5,12), (5,13), (5,14), (5,15), (6,12,0.5), (7,12), (8,12), (9,12), (10,12), (11,12), (12,8), (12,9), (12,10), (12,11), (12,12)
Note: See obstacle_problem2.py on Canvas for obstacle list typed in already. This should save you from any copy-paste errors.

An example of the required plot is shown below, although this is just an example and is not representative of this exact problem.



4. **Super TSP - Brute Force** You have a map very similar to Problem 2 (just a few obstacles removed), with many more waypoints that must be visited. The vehicle **must** start from the first delivery point (1,1). Use your A*-based TSP code to identify the best path. Plot identical to Problem 2. Please note this will take 10 minutes up to 60 minutes to compute (based on your computer performance). It is recommended to use the tqdm package to keep track of timing. Simply put "from tqdm import tqdm" at the top, and in your big for loop put "for i in tqdm(range(0,len(perm_list))):" 13%

   Grid Info: Xmin = 0, Xmax = 15, Ymin = 0, Ymax = 15, Grid Size = 0.5, Robot Radius = 0.5
   Delivery Points: (1,1), (9,7), (1,9), (4,4), (9,4), (6,14), (3,11), (14,1), (1,14), (14,14), (7,10)

   Note: See obstacle_problem3.py on Canvas for obstacle list typed in already. This should save you from any copy-paste errors.

   Provide the computation time, and the minimal total travel cost.

5. **Super TSP - Genetic Algorithm** Use the same map/obstacles as Problem 3. You must use a Genetic Algorithm to solve the TSP. It is recommended to use/adapt the code provided on Canvas. It is recommended to comment the code (as you need to understand the basics of how it is working to make the changes). There are several ways to approach this problem, but keep in mind you must start at the first "city" or waypoint. The code currently just provides the best path (can start at any city). It is recommended to remove the first city from the list, and add it in whenever computing the current path cost (or fitness). This is an easy solution that doesn't require a lot of modifications in the code. 17%

   Use a population size of 500, with 2,000 generations.

   Provide the computation time, and best path cost.

   Provide a plot of the obstacles with path (identical to previous problems).

Question:	1	2	3	4	5	Total
Percent:	15	30	25	13	17	100
Score:						