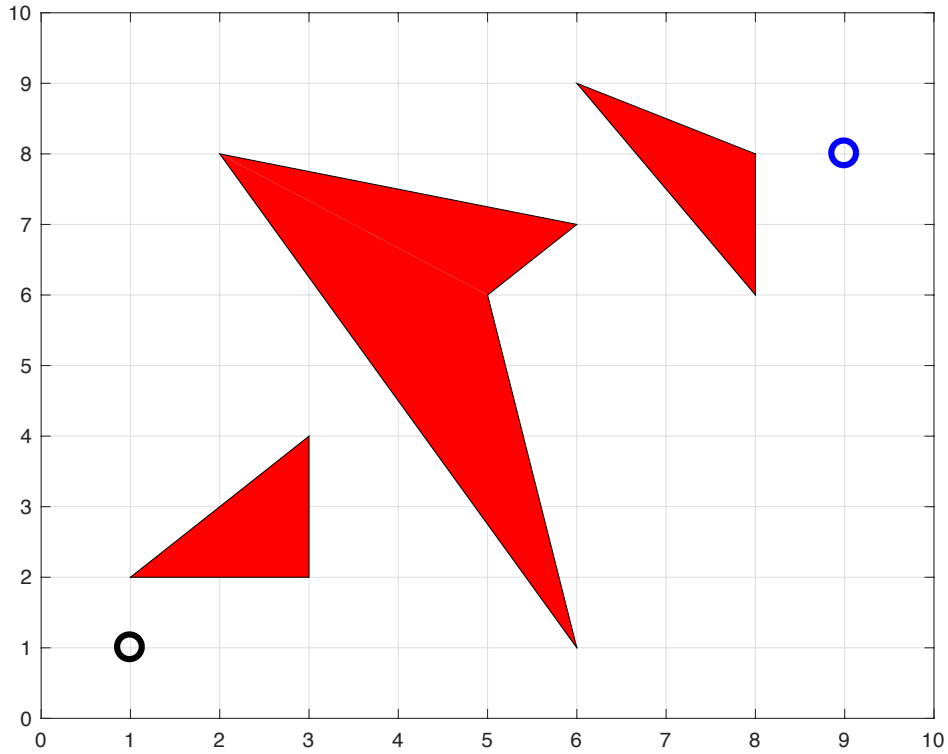
ME 401/5501 – Robotics and Unmanned Systems
HW #1: DUE August 31st, 2022

LATE HOMEWORK WILL BE PENALIZED 10% PER DAY

Problem 1:

Using the map shown below, generate the visibility graph (include the start and end nodes). Additionally, show the reduced visibility graph with a different color (i.e. blue for reduced graph, black for remaining standard edges). You do **not** need to compute the edge costs.

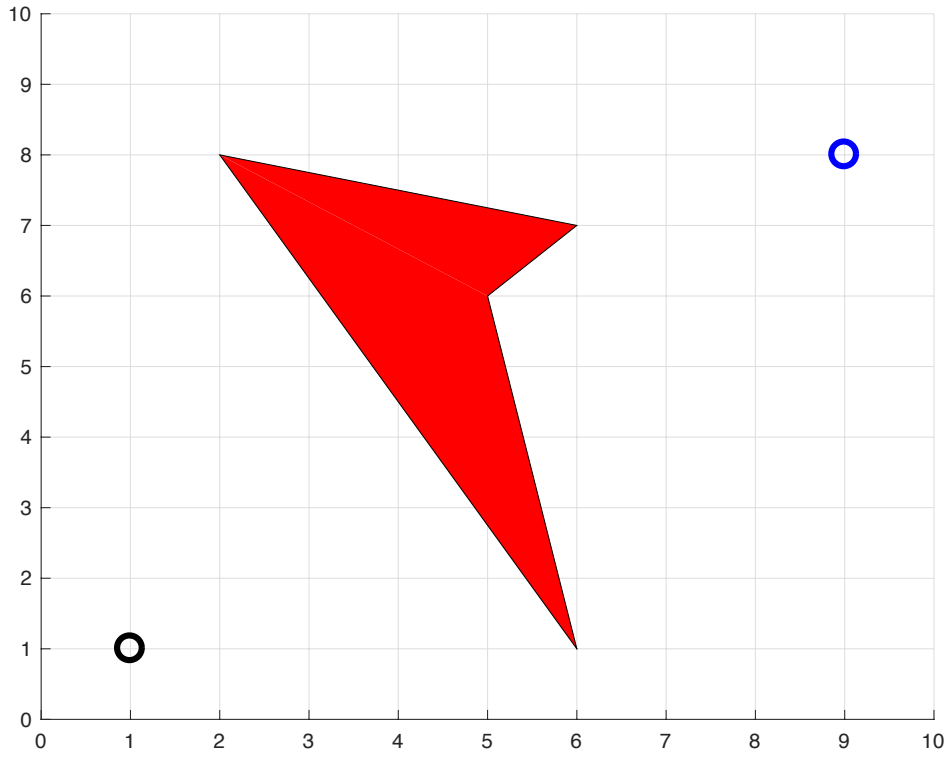


Problem 2:

Using the Python Class tutorial available at https://docs.python.org/3/tutorial/classes.html, create a class called **node** that has the following instance variables, **x**, **y**, **parent_cost**, and **index**. Provide your short Python script that contains this class.

Problem 3:

Using the map shown below, show the reduced visibility graph along with the Euclidean distance for each edge. Highlight the shortest path from the start (1,1) to the goal (9,8).



Problem 4:

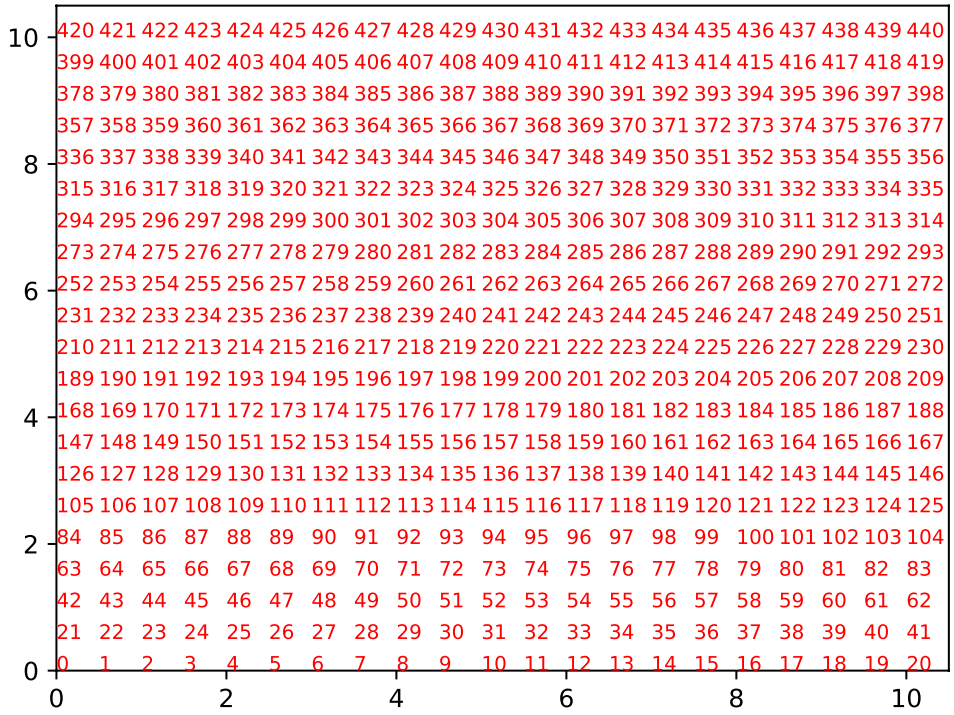
Given the following map parameters, generate a figure similar to the one shown in which you are computing each node index and plotting the index at the corresponding node location. The node index is simply the unique name/value associated with the node. You need to write your **Python** script such that any node location (x and y pair) returns the node index. I.e. simply making counter that plots at each node location will not work. Use the equation(s) developed in class to assist in this problem. This node index is crucial in generating grid-based path planning techniques. A small function that computes the node index is an efficient method for computing the index.

Grid Spacing = 0.5
Min X = 0 Max X = 10 (include both 0 and 10 in your grid)
Min Y = 0 Max Y = 10

Notes:

When generating the x and y values (that span from 0 to 10), the NumPy command *arange* is particularly useful. Make sure that you capture the end point by adding an extra "grid_size" onto the end value.

Matplotlib.text(x, y, str(int(number to display)), color="red", fontsize=8) is a good function to use to stick the text in the figure.



Problem 5:

Create a function that calculates the distance from one **node** to another. Pass two nodes to the function and return the Euclidean distance. Test your function by having it calculate the distance from (2,1) to (3,2). Make sure the answer is correct.

Submit your Python code.