Academic Honesty Statement

Academic honesty and plagiarism:

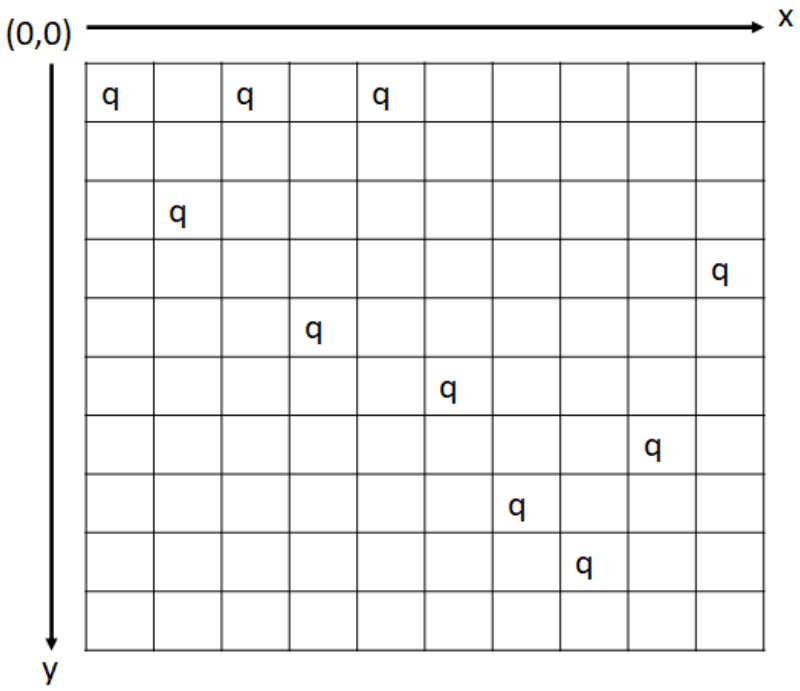
- Plagiarism is unacceptable.
  - Code you submit must be your own. No copying, adapting, or submitting code you did not create is allowed. Working together and presenting variants of the same file is not acceptable. Here are some specific guidelines to make sure you don't cross the line:
    - Do not exchange programs or program fragments in any form on paper, via e-mail, photos, or by other means.
    - Do not copy solutions from any source, including the web or previous quarters' students.
    - Do not discuss code with other students at the level of detail that will lead to identical programs or program fragments.
    - Do not use chatGTP or other AI tools
  - Ask me if you are uncertain.
  - All violations of the academic honesty will be immediately referred to the dean's office.

Lab 2: Greedy Search
Due April 20, 7pm

The task in this lab is to use greedy hill-climbing search to find minima and solutions to the 10-queens problem. This problem tries to find a position for all 10 queens on the board where they cannot attack each other. Reminder, an attack is whenever a queen shares a row, column, or diagonal with another queen. Here we will provide an initial configuration of the board for the 10-queens problem, and you are to modify that board with the local minima reached from that starting board. Here, a single step in the hill-climb is the movement of any one of the queens.

To simplify the problem every initial configuration will have exactly one queen in each column. We will also require that any intermediate configuration used also has exactly one queen in each column. This means any step in your hill-climb is the motion of one queen vertically up or down within her column.

Each board will have the following layout of the x and y coordinates:



The board will be given as a 2D array board[y][x], where x and y are the positions of the board. Queens are marked with a 1, empty positions are marked with a 0. You are to complete the code for the function **gradient_search**(board) in the file student_code.py. This function will modify "board" to be the local minima reached. In addition you will return True if the local minima is also a solution to the problem, and return False otherwise.

To make sure everybody arrives to the same results (very important for the automated grader) you **must** use the following tie breaker:

If motion of two or more different queens give the same best reduction in total attacks, choose to move the queen with the lowest x value. If two or more positions for the same queen give the same best reduction in total attacks, choose the position with the lowest y value.

Do not move to a new board state unless the total number of attacks are reduced (i.e. don't move on ties)

Don't use any additional python modules or packages, and use Python3.

Considerations:

- We provided several maps to let you test your solution, but the grading will use a different set. Feel free to create your own test cases.
- The running time of your algorithm cannot be longer than 10 seconds per board, otherwise it will fail the grading test.
- Queens are considered to have an attack between them even if there is another queen between them. For example 4 queens in the same row will count as 6 attacks.