CSC 4444: Beyond Search (200 pts)

Adapted from: Francisco Iacobelli

1 TASKS

In this assignment you will compare two search algorithms to solve the 8-queens problem. You will be evaluated on 3 components to this assignment:

1. Design documentation (Submit 1-week before final code deadline)
2. Quality and thoroughness of code comments
3. Code solving the search problem

1.1 Documentation and Code Comments (100 pts total)

- **Link to Resources Document**
  - Document contains links to examples and explanatory articles for both documentation and code commenting.

- **50 pts: Design Documentation**
  - Audience & Genre:
    Your audience is a fellow student who needs to complete this assignment. *Your documentation should give them all the information and basic frameworks they will need to implement the code and evaluate its performance.* The genre is design documentation.

  - "A design doc — also known as a technical spec — is a description of how you plan to solve a problem" (link). Most articles and examples of design documentation you will find describe these docs in the context of software development; however, the scope of the problem you are solving here is *much* smaller. You will apply the basic components of a design doc to this assignment, scoping them to be appropriate for this code.

  - Your document should include *at least* the following:
    - Title and author
    - Overview of the problem
    - Design plan (may include pseudocode and/or diagrams - this is where you will describe the logic necessary to solve this problem). In this section, describe *how* the algorithms you selected can be used to solve this problem.

        - Alternative approaches to solving this problem (Why did you choose these algorithms for this problem? What other algorithms could you have used?)
        - Keep in mind: another student reading this doc should understand the problem and a basic framework (algorithms, basic logic, etc.) for implementing a solution.

    - **Submissions**
      Because this documentation details how you will solve the problem, it should be completed early in the process. You will bring a draft version to class for peer review and then submit your final version one week before the code is due.

      I will assign an appropriate grade based on clarity and completeness.

      1. **Draft:** bring to class for peer-review on Thursday 10/6 (will count as classwork)
      2. **Final version:** submit by Wednesday 10/12

- **50 pts: Code Comments & Good Variable Names**
    - Read these articles (link1, link2, link3) to understand what good code commenting looks like, and why it is important.
    - Use functions to compartmentalize your code, and ensure that every function is preceded by a comment that explains what it does.
    - Add other comments within your code to explain sections with complex logic or anything else that is not easily understood by reading the code alone.
    - If you have questions about what to do here - be sure to ask.

1.2 Comparing Searching Algorithms - Code (100 pts total)

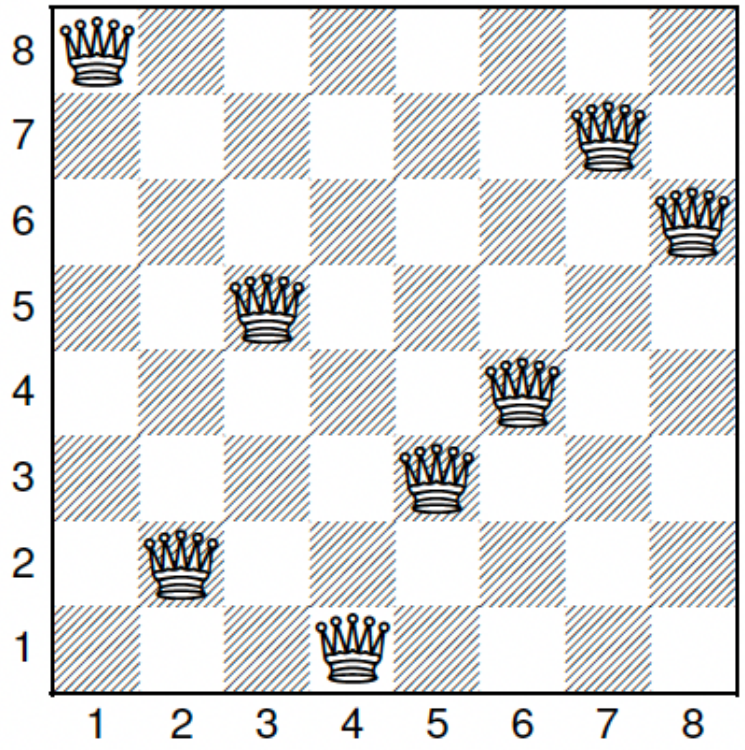
Create a python program called 8-queens.py that offers the user two options:

1. Visualize the search and solution for a random 8-queens board
2. Enter an integer indicating how many random 8-puzzle boards to generate and solve (note that some algorithms may sometimes get stuck).

Follow the directions below:

1. **10 pts:** The program should ask the user if they want to visualize the search of a solution for one board (see Step 5 below), and if not, how many boards they want to

generate. Each puzzle should be represented by a list with 8 elements. Each element indicates the position of a queen in a column. For example: [8, 2, 5, 1, 3, 4, 7, 6] represents the following board:



2. **40 pts:** In 8-queens.py, there should be at least two functions that implement two different algorithms to solve a puzzle (don't forget to compute the heuristic).
   -standard hill-climbing
   -hill climbing with random restart
   -genetic algorithms
   -simulated annealing.
   (+10 extra credit points each for using genetic algorithms or simulated annealing)

3. **15 pts:** Each function should also measure the search cost (number of boards generated until you hit a local/global minimum) and whether the problem was solved.

4. **25 pts:** At the end of the execution of 8-queens.py (generating x random boards - where user inputs integer x) there should be a message with:
   The number of problems tried, percentage of solved problems using each algorithm, and average search cost of each algorithm.

If you use random restart, print either the average number of restarts required to find a solution, OR the percentage solved if you use a fixed number of restarts.

A program testing hill-climbing and simulated annealing may output something like[1]:

```
350 puzzles.
Hill-climbing: 15% solved, average search cost: 2489;
Sim. Annealing: 97% solved, average search cost:1453;
```

5. **10 pts:** Allow the user to choose to visualize a board search. For this option the program should:
    a. Follow the same logic as above, except it will only try 1 board (think about how you can use the function we wrote in class for visualization, and just make a small adjustments to your logic in main to achieve this)
    b. Display the algorithm selected, starting board and heuristic of the board.
    c. Display the successors and final board, number of steps to the final state (include the heuristic if this is not a solution).

---

[1] I made up the numbers. Your numbers will be different

2 MUST HAVE

- You must submit your design document as a PDF
- You must have only ONE python file called 8-queens.py. It should ask the user whether or not they want to visualize solving one board, or for a number that indicates the number of random puzzles to test.
- You must use Python 3.x, and include the following:
    - The second line should have the name of the assignment and optionally a couple of words about it. These should be enclosed in three quotation marks. For example: """ Eliza homework. Relationship advisor """