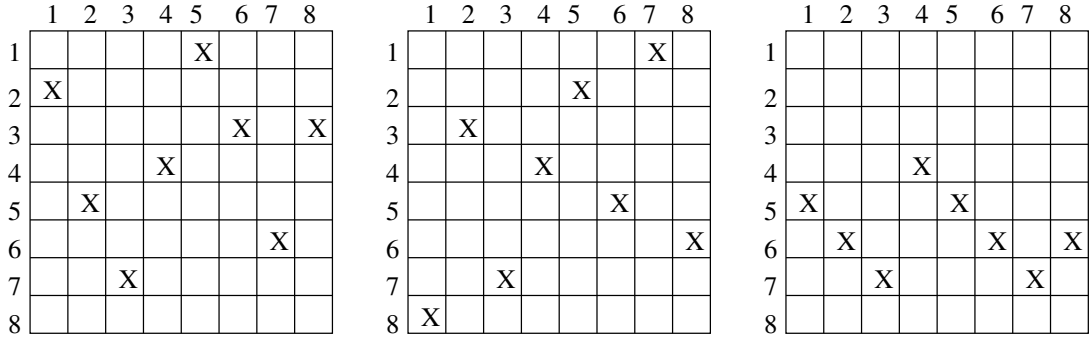
CS 771: Artificial Intelligence

Programming Assignment, Spring 2021: Due Fri, April 16

100 Points = 20% of Overall Grade

You will write a program to solve the 8-queens problem. The input to the program is a start state. The ouput is the sequence of states your program goes through; the *first state* in the output sequence is the start state; the **hope** is that the *last state* in this sequence provides a conflict-free solution to the problem.



Input and output of states: A state $s = (Q_1, Q_2, \ldots, Q_8)$, where Q_i is the row number of the queen in column i. It is input or output as a sequence of 8 digits, where each digit is between 1 and 8; the digits are separated by **single** blanks, with **NO** blank at the end. So, the input/output of a state takes exactly 15 spaces (8 digits plus 7 blanks). Each state should be output on a separate line, with the first digit appearing in column 1 (of the output file). For example, the states in Figures a, b and c, respectively, are:

2 5 7 4 1 3 6 3
8 3 7 4 2 5 1 6
5 6 7 4 5 6 7 6

Choosing the next state: For each queen, see how many other queens it attacks. Pick the queen Q with the largest number of attacked queens (say number of attacks is n_1). If there are several such queens, then you **must** pick the one with the smallest column. Now, consider moving this queen Q within its column to the row that minimizes the number of queens it attacks (say number of attacks is n_2) . If there are several such rows, you **must** pick the smallest row; move Q to this row. This gives us our next state.

The process in the above para is repeated until we reach a state in which $n_1 = 0$ (success), or for all queens with n_1 attacks, n_2 is $\geq n_1$ (failure).

Special cases: If $n_1 = 0$, we have a solution, and we stop.
If $n_2 = n_1$, then we do not move this queen Q; we look for another queen Q' that attacks n_1 queens; if there are several such queens, we pick the one with the next smallest column. If for all queens with n_1 attacks, n_2 is $\geq n_1$, we stop our search.

Checking if two queens attack each other: For $i \neq j$, Q_i and Q_j attack each other if either $Q_i = Q_j$ (same row), or $|Q_i - Q_j| = |i - j|$ (same diagonal).

Program Run: The input file has several start states, one per line. Your program should have a loop that starts with each of those start states.

For example, consider the following input file with three *start* states:

2 5 7 4 1 3 6 3
8 3 7 4 2 5 1 6
5 6 7 4 5 6 7 6

For start state 2 5 7 4 1 3 6 3 (Fig a): Q_6 and Q_8 have the most number of attacks, which is 1; so $n_1 = 1$. By our tie-breaking rule (pick the queen with the smallest column), we pick Q_6 to move. Moving

Q_6 to row 8 (within its 6th column) results in minimum of attacks $n_2 = 0$. So, we must move Q_6 to row 8; the next state is $(2, 5, 7, 4, 1, 8, 6, 3)$.

In this resulting state, most number of attacks for any queen is $n_1 = 0$. So, we stop with success.

For start state 8 3 7 4 2 5 1 6 (Fig b): Q_4 and Q_7 have the most number of attacks, which is 1; so $n_1 = 1$. By our tie-breaking rule (pick the queen with the smallest column), we pick Q_4 to move. But moving Q_4 to any other row (within its 4th column) results in a minimum of two attacks; i.e., $n_2 = 2$. Since $n_2 \geq n_1$, we decide to not move Q_4.

So , we next pick Q_7 to move. But moving Q_7 to any other row (within its 7th column) results in a minimum of two attacks; i.e., $n_2 = 2$. Since $n_2 \geq n_1$, we decide to not move Q_7.

Since we have considered all queens with n_1 attacks, we stop (failure, since $n_1 > 0$).

For start state 5 6 7 4 5 6 7 6 (Fig c): The number of attacked queens are $(3, 5, 4, 4, 5, 5, 5, 3)$. So, $n_1 = 5$; we need to consider Q_2, Q_5, Q_6, Q_7 for move, in that order. For Q_2, moving it to rows 1 or 3 gives minimum number of attacks $n_2 = 0$. So, we must move Q_2 to row 1; the next state is $(5, 1, 7, 4, 5, 6, 7, 6)$. Starting with this resulting state, compute the next state, and so on.

Overall, on the above input file, the output file looks like:

```
2 5 7 4 1 3 6 3
2 5 7 4 1 8 6 3
Success

8 3 7 4 2 5 1 6
Failure

5 6 7 4 5 6 7 6
5 1 7 4 5 6 7 6
     ⋮
5 1 7 4 2 3 8 6
Failure
```

In your output file, "Success" and "Failure" must appear exactly as shown above, starting in column 1, with no blanks at the end.

What to submit : You can write your program in any language you want. Submit:

- All files that are necessary to run your program . Name and WSU ID must appear on top of each file that you submit.
- A README.txt file that explains very clearly how to compile/execute your program. Without this README file your program will not be graded; you will get 0.

Late Submissions: Assignments after deadline will be accepted upto 3 days late. After that they will be rejected. Late penalty: 10 points per day.