ECS 170: Spring 2022
Homework Assignment 1

Due Date:

No later than Thursday, April 14, 2022, 9:00pm PDT. Note that your next homework assignment may be posted before then. **You are expected to do this assignment on your own, not with another person.**

It is possible (maybe even likely) that we will have to make some adjustments to this assignment as the days go by. Please try to be flexible.

The assignment:

Consider the peg puzzle program we have discussed in class, which is written in Python and has also been posted to Canvas as `pegpuzzle.py`. Your task is to adapt this program so that it solves an arbitrary eight-tile puzzle. Call your new program `tilepuzzle.py`.

Just like the peg puzzle program, your tile puzzle program will be invoked with any valid start state (the first argument) and any valid goal state (the second argument). Again, just like the peg puzzle program, your tile puzzle program should return a list of states representing a path from the start state to the goal state.

For example, the state in the tile puzzle that looks like this:

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

will be represented by the following Python list of lists:

```
[[2,8,3],[1,0,4],[7,6,5]]
```

Note that the integer `0` represents the empty tile or space.

When your tile puzzle program finds a path from the start state to the goal state, it should return a list of states representing that path, beginning with the start state and ending with the goal state. For example, when your program is called with these three arguments:

```
>>> tilepuzzle([[2,8,3],[1,0,4],[7,6,5]],[[1,2,3],[8,0,4],[7,6,5]],9)
```

your program should **return** (not print) a list that looks like this:

```
[[[2,8,3],[1,0,4],[7,6,5]], [[2,0,3],[1,8,4],[7,6,5]],
 [[0,2,3],[1,8,4],[7,6,5]], [[1,2,3],[0,8,4],[7,6,5]],
 [[1,2,3],[8,0,4],[7,6,5]]]
```

Your program does not have to find this particular path. The path shown above is only one example of the many solutions your tile puzzle program could find. Your program need only return the first solution it finds, not the most optimal solution.

Notice that there is a third parameter in the tile puzzle program that was not included in the demonstration in class. What's that extra parameter? That parameter allows you to pass an argument that serves as a limit to the depth of the search so that the program doesn't do a lot of unnecessary time-consuming recursion. You must add that parameter to the program and use it appropriately. If the argument passed through that parameter is 9, for example, then the program must not search more than 9 levels deep.

When adapting the peg puzzle program to solve tile puzzles, you must change the name of the top-level function to `tilepuzzle`. You will also need to add the third parameter to this function and to the `statesearch` function. Those are the only changes you may make to the first two functions in the file `pegpuzzle.py`.

If you want your program to use a different representation for the tile puzzle internally, that is fine with us as long as the program accepts arguments and returns results in the exact form described above. (In other words, you would translate our representation to yours on input, and then translate your representation to ours prior to returning the result.)

Your program should be written using Python 3, version 3.6 or later. You should not be using Python 2 in this class.

Motivation:

This is an easy homework assignment. So why do this? First, it gives students who are using Python for the first time an opportunity to practice those new skills. Second, it gives students who have some Python experience a chance dust off those old skills. Third, and most important, it gives everyone the opportunity to work on representing states, constructing operators, and applying operators to a state to generate new states. These are all skills that may be useful in the very near future.

Turning it in:

Submit your program as a text file named `tilepuzzle.py` via Canvas.