Instruction:

For this assignment, you need to implement an Iterative Deepening search (IDS) and A-star algorithms to solve a puzzle problem. For this puzzle game, the GUI (graphical user interface) framework is provided that is developed by Python and **pygame library**. Below is a highlight of the provided framework.

Two folders are given:

1. **game** folder - This folder deals with all the graphical or visual stuff, including pattern generation from an image, user mouse/keyboard interface, etc (**You do not need to modify any files in this class**).

2. **sol** folder - This folder has the file that provides a solution for a given puzzle. Two search functions you can find from this file "iterativeDeepening(puzzle)" and "astar()". These search functions are triggered automatically when the user clicks the "IDeep" or "A Star" button in the GUI accordingly.

What you need to do is to create Python project (suggested using the Interpreters 3.0 or higher) and copy these two folders directly to the created project. Then you need to install the **pygame** library, e.g. pip tools. Open the "puzzle.py" file in the game folder and run the code from there.

Requirements:

You need to implement the two functions in the "solution.py" file in the "sol" folder:

```
def iterativeDeepening(puzzle):
    list = []
    return list
```

and

```
def astar(puzzle):
    list = []
    return list
```

The input argument "puzzle" is just a list of 9 numbers, e.g. [4, 3, 5, 2, 0, 1, 8, 6, 7], where the number "8" represents the empty space. You can use a print() function to see this input data. Both functions expect you to return a list of a path for the number "8" to move to make the sequence into a sequential order. Each element of the path represents a new position of the number "8" to move. The details are explained in the section below.

What should be submitted:

You can only submit the "solution.py" file, since you only implement this function. Of course, if your code involves modifying other files (not encouraged), you can submit other files together by zipping them. If your code changes any other files, please provide a document describing what other files are involved in changes..

Explanation of the returned “path” list:

The path stores the moving steps of the empty space “8" to move though when you play the game you move the tiles rather than the empty space. Below shows an example of the moving path of “8”

For example:

Input: data = {0, 4, 1, 3, 8, 2, 6, 7, 5 };

Goal: make it into the correct order {0, 1, 2, 3, 4, 5, 6, 7, 8}

You need to make the following changes on the number 8, since the number 8 represents the empty space, moving 8 to its neighboring numbers equals to moving the corresponding number to the empty space. Below it shows a demo of the steps:

0 4 1 swap with 4	0 8 1 swap with 1	0 1 8 swap with 2	0 1 2 swap with 5	0 1 2
3 8 2	3 4 2	3 4 2	3 4 8	3 4 5
6 7 5	6 7 5	6 7 5	6 7 5	6 7 8

So from this example, the right path should be {1, 2, 5, 8}.

WHY? You may have thought it was {4, 1, 2, 5}, since the number 8 has swapped with them in this order. That is true. However, we do not care which number it swapped with, but which position the number 8 has gone through. As the numbers can be in any positions during different time, which give no hint about where the number 8 is. In contrast, the positions are fixed. So we assume the positions are in the same order as an increasing sequence:

	[0]	**[1]**	**[2]**
Fixed Position =	**[3]**	**[4]**	**[5]**
	[6]	**[7]**	**[8]**

Here, I use "[]" to distinguish the positions from the numbers. So now you can see, the number 8 starts from position [4], then swapped with number 4, which goes to the position
[1]; then swapped with number 1, which goes to the position [2]; then swapped with number 2, which goes to the position [5]; finally, swapped with number 5, which goes to the position [8]. So the path you should assign to the parameter "&solution" with the path sequence {1, 2, 5, 8}.

Rubric:

(1) Your code is free of compilation error or runtime errors (10%)
(2) For the iterative deepening search, your code can iteratively increase depth limit and perform DFS correctly (10%)
(3) Successfully return an optimal path by iterative deepening search (10%)
(4) Correctly define Heuristic function h() for A-star search (10%)
(5) Correctly compute the total cost f(S) = g(S) + h(S) for each move (10%)
(6) A-star search correctly expand states in the right order (15%)
(7) A-star search correctly get an optimal path (25%)
(8) Provide a short document describing what you have achieved and limits, with some snapshots (10%)