Artificial Intelligence

Lab

Lab Instructor

Shafaqat Ali

Programming Assignment # 1

Submission:

- Submit your work in a single zip file with the name **StudentName_RollNumber_01.zip** containing **code and report.pdf**
- The assignment is only acceptable in a jupyter notebook python file with **.ipnyb** extension. No PyCharm, Spider, or other IDEs.
- You will submit a report containing the screenshots of your output(program output) and any other detail if you want to provide.
- Follow the above naming convention for submission, there is a 5% penalty if you don't follow it.
- 10% (of obtained marks) deduction per day for a late submission. There will be no submission after 5 days.
- You cannot look at others' code or use others' code, however, you can discuss it with each other. Plagiarism will lead to a straight zero with additional consequences as well.

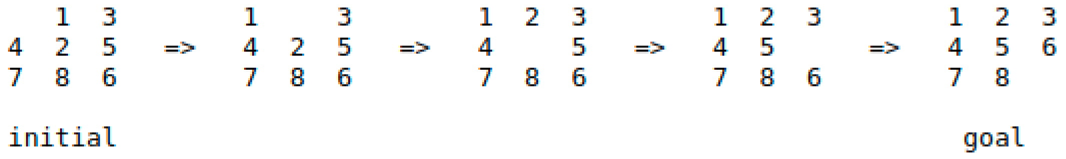
Due Date : 15/05/2021

Note: For this assignment (and for others in general) you are not allowed to search online for any kind of implementation. Do not share code or look at the other's code. You should not have any implementation related to the assignment, other than your own.

Programming Task

Write a program to solve the 8-puzzle problem using the A* search algorithm.

The 8-puzzle problem is a puzzle invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square labeled with 0. Your goal is to rearrange the blocks so that they are in order. You are permitted to slide blocks horizontally or vertically into the blank square. The following shows a sequence of legal moves from an initial board position (left) to the goal position (right).



```
   1  3        1     3        1  2  3        1  2  3        1  2  3
4  2  5   =>   4  2  5   =>   4     5   =>   4  5      =>   4  5  6
7  8  6        7  8  6        7  8  6        7  8  6        7  8

initial                                                      goal
```

Calculating Cost at any Level:
So far, we were calculating the cost **f(m)= g(n)+edge_cost_from_n-to-m,** but here for simplicity during implementation you will consider that the cost for all edges is 1 so that you can ignore it.

The cost will be calculated as the number of mismatched tiles between any current state and goal state.

Implementation Details:

- Create a class named **Puzzle**
- Create a constructor and Initialize necessary variables(i.e. lists, queues or dictionaries etc).
- Create a function **h_mismatch_cost(states...)** which returns the cost of current state.
- Create a function **a_star_traversal()** which takes initial state as input and traverses from initial state till goal state.
- Create one or more than one functions to check if movement of blank tile in a certain direction is feasible and for creation of state after moving blank tile.
- Create a function **display_path()** which displays the path from initial state to goal state as shown in above figure.

🤓 Good Luck 🤓