Assignment

- Title: A* algorithm

- Problem statement:
  Solve 8-puzzle problem using A* algorithm any initial configuration and define goal configuration clearly

- Objectives:
  - The learn and understand use and need of A* algorithm.
  - To apply A* algorithm to real time problem
  - To implement A* algorithm using suitable programming language

- Outcomes:
  We will be able to
  - learn about A* algorithm
  - apply A* algorithm to gaming problem
  - implement A* algorithm using Prolog / Python / Java

- Hardware & Software Requirements:
  - OS: Fedora 20 / Ubuntu (64-bit)
  - RAM: 4 GB
  - HDD: 500 GB
  - Eclipse IDE
  - Java JDK v1.8
  - Python libraries.

- Theory:

  - A* is one of the most popular heuristic search engine for finding paths in graph.
  - It is really a smart algorithm which seperates it from other conventional algorithm.

  - Consider a square grid having many obstacles and we are given a starting cell and target cell.
  - We want to reach target cell from starting cell as quickly as possible
  - What A* algorithm does is at each step, it picks the node according to a value '-f' which is a parameter equal to sum of other two parameters -g & -h.
  - At each step it picks the node cell having atleast '-f' & process that node/cell

  - We define 'g' & 'h' simply as possible
    g = the movement cost to move from the starting pt. to a given square on the grid following the path generated to get there.
    h = the estimated movement cost to move from that given square on grid to final dest. this is often refered to as the heuristic which is nothing but a kind of a guess.

• Algorithm:

```
1.) Initialize the open list
2.) Initialize the closed list
        put starting node on the open list
3.) While open list is not empty.
        (i) Find the node c̄ the least f on the
            open list. Call it 'q'
        (ii) pop 'q' off open list
        (iii) Generate 'q's successors
        (iv.) for each successor
            (a) if successor is the goal, stop
                successor.g = q.g + distance(successor.q
                successor.h = dist from goal to successo
                successor.f = successor.g + successor.h
            (b.) if a node with same position
                as successor is in open list which
                has lower 'f', skip this successor
            (c) if a node with the same position
                as successor is in the closed list
                which has a lower 'f' than
                successor, skip, otherwise, add
                node to open list
        (v.) end for
        (vi) push q on closed list
4.) end while
```

- Test cases:

Initial Configuration			Final configuration		
1	2	x	1	2	3
4	5	3	4	5	6
7	8	6	7	8	x

Output :
The puzzle was solved in 18 moves

- Conclusion:
We successfully implemented A* algo for 8puzzle problem.