**Programming Project 1**

8-Puzzle problem using A\* heuristic Algorithm

**8-Puzzle problem** – This is a 3-by-3 puzzle wherein there is a grid with 8 square blocks labeled 1 through 8 and a blank square. The aim of puzzle is to attain a certain given configuration (goal - used in code as a term to identify the desired configuration) by rearranging the blocks. You are permitted to slide tiles - Up, Down, Left and Right into the blank space.

**Algorithm used** – We are using A\* Misplaced tiles and Manhattan Distance heuristic to determine the expanded node, the path, its cost and the depth.

**Program Structure**- We have created two Java files i.e. Final and Node. Node store information of current state, parent, cost, depth and count whereas Final has the main algorithm. The program ask user to enter the initial state and the desired goal for the puzzle and the type of heuristic user wants to solve the puzzle with.

* Array List (expandedNode) and Priority Queue (fringe) is used in our algorithm to store the expanded array so that we can compare current states with the already visited states and popping out the child based on the minimum cost from priority queue.
* Functions used –
* copying() – to copy the initial state to the child and then apply all the other function to it.
* position() – to find the position of the 0 i.e. blank
* movingTile() – to generate child of the current state based on the position of 0 i.e. blank. It checks whether this new generated state is already there in the Array List or not. If it is False, the state is then added to both Array List and the Priority Queue otherwise it is being ignored and same steps continue for every child generated.
* checkingState() – where we are comparing new child with already visited node in the Array List and returning boolean value for it.
* heuristicMisplacedTile() – to find the heuristic using the Misplaced Tiles
* heuristicManhattanDistance() – to find heuristic using the Manhattan Distance.