8 puzzle solution- TPEC Module-1 Assignment

- We include the C programming built in library functions for input/output and limit such as stdio.h, stdlib.h, limits.h
- We are considering a 3x3 puzzle block so we use N to declare the dimensions

  MAX_STATES is the number of possible states.

- PriorityQueue contains node in the priority order. PQNode represents a node in this particular queue.
- State tells the position of the board and blank spaces in the board.
- Initially a start state of the board is being initialized to get a goal state of the puzzle using priority queue.

- Hamming priority function- distance of the board and the number of moves made so far to get to the search node. a search node with a small number of tiles in the wrong position is close to the goal, and we prefer a search node if has been reached using a small number of moves.
- The Manhattan priority function is the Manhattan distance of a board plus the number of moves made so far to get to the search node.

- isGoalState checks whether the states that are obtained are the final goal stage or not.
- We are using functions like calculateHammingPriority and calculateManhattanDistance to determine the priority functions in the solution and moves made in the board.
- calculateManhattanPriority helps to calculate the sum of distance between blocks in the board.

- Swap helps in swappin of the blocks in the puzzle in order to change their placements.
- printBoard executes the current or present state of the puzzle.

- Push adds a state to the priority queue with a given priority. Pop removes and returns the state with the highest priority from the priority queue.
- isEmpty` checks if the priority queue is empty.

- A* search- We define a *search node* of the game to be a board, the number of moves made to reach the board, and the previous search node.
- aStarSearch performs the A* search algorithm to find the solution. It starts from the initial state, and explores all the possible state states, calculates priorities for each state based on heuristic functions, and continues until it reaches the goal state or all possibilities.

- The code uses malloc for dynamic memory allocation.
- The main function initializes the initial state with the given puzzle configuration. It checks if the initial state is already the goal state.

- That is if the generated puzzle is already in the solved state or not.
- If not, it prints the initial state and calculates the Hamming and Manhattan priorities.
- The aStarSearch function is then called to find the solution.
- If the goal state cannot be reached, it prints a message indicating that the goal is not reached, else it prints the initial state, hamming and manhattan priority and steps to reach the goal plus the goal state of the puzzle.