This code implements an A* search algorithm to solve the 8-puzzle problem, a sliding puzzle game. The puzzle board is represented as a 3x3 matrix, where the goal is to arrange the tiles in ascending order with the blank space (0) in the last position.

The `State` structure holds the current board configuration, the position of the blank tile, and the cost (number of moves from the initial state). The priority queue (PQ) is used to manage and explore states based on their priority, calculated using the Manhattan distance heuristic.

The `initializeState` function sets up the initial state from a given board. The `isGoalState` function checks if the current state matches the goal configuration. Priority calculations are handled by

`calculateHammingPriority` and `calculateManhattanPriority`, which measure how far the current state is from the goal state.

The `aStarSearch` function performs the A* search. It repeatedly explores the state with the lowest priority, generates successor states by moving the blank tile, and pushes these states into the priority queue until the goal state is reached or the queue is empty. The main function initializes the initial state, checks if it's already solved, and then starts the search while displaying relevant information.