Israel Alcantara

Problem 1

I structured the solver around three main phases, each expressed in its own set of routines. First, the movement & swap, filter, and expand functions handle the raw state transitions. I reused the tile‑shifting code from GenGemPuzzle.c to move the blank (0) up, down, left, or right. Each move first guards against falling off the 4×4 board, then swaps the blank with its neighbor and updates the blank's coordinates. The filter routine then walks either the open or closed list, comparing each node's board to newly generated successors. If a duplicate is found, it frees that node and nullifies the successor pointer. In expand(), I clone the current node four times, apply each move, and discard any clone that leaves the board unchanged, linking survivors back to their parent for later backtracking.

Next comes the heuristic update and open-list merge. In update_fgh(), each valid successor increments its cost‑so‑far (g) by one and recomputes the estimated remaining cost (h) via the sum of Manhattan distances of all tiles from their goal positions, the total score f = g + h then guides A*. The merge_to_open() function keeps the open list sorted by f, walking down the linked list until it finds the right insertion point (or resets the head if the list is empty or the node has the smallest f), ensuring that the next node expanded is always the most promising.

Finally, the main search loop ties everything together. After validating exactly 16 integer arguments and calling initialize(), I repeatedly remove the head of the open list, check it against the goal, then call expand(), filter(), update_fgh(), and merge_to_open(), moving each expanded node into the closed list. When the goal is found, the code traces back via parent pointers to build a solution path and prints each board in start-to-goal order. If the open list ever empties without success, it prints no solution. I verified correct behavior under three conditions, initial equals goal, a shallow scramble (GenGemPuzzle 4), and a deeper scramble (GenGemPuzzle 40), with each case finishing within the one-minute requirement and producing clear, step-by-step output.