Maria Angel Palacios Sarmiento

In order for the 15-puzzle program to properly represent the initial and final states and produce legitimate moves from empty space (represented by 0), I had to modify and implement a number of crucial functions. I started by creating the initialise function, which sets the target state with 0 in the lower right corner and creates the board's initial state using data from the command line. Along with the swap function, which switches values in the array, I also constructed the movement functions (move_up, move_down, move_left, and move_right) that enable the position of 0 to be switched with its neighbours. These functions are employed to create child states from the current state during node expansion. Furthermore, based on a direction preference heuristic that depends on the location of the zero, I built the expansion logic in the expand function, which creates up to four successor nodes from the current node. I also created the update_fgh function, which uses the Manhattan distance as a heuristic (h) to update the f, g, and h values of each node in order to optimize the search. The created nodes are then inserted into the open list by the merge_to_open function in order of their f value (priority), enabling more effective growth.

In order to reduce runtime, I lastly developed the filter function, which compares new nodes with those that already exist in the open or closed lists and eliminates superfluous duplicates. We can now approach a more structured and effective solution to the problem because of these advancements. Also now, the program marks it as unsolvable when there are more than 1000 iterations.