This document will explain the g(n) function and h(n) functions chosen for my heuristic function in this assignment. Additionally, the document will explain why the heuristic function is admissible.

The equation for g(n) is as follows:

$$g(n) = g(n)\ of\ the\ preceding\ state + value\ of\ the\ swapped\ tile$$

This accounts for the cost of the states leading up to the new state and the cost of the move itself, which is the value of the swapped tile. As the moves progress, the g(n) cost of each state increases appropriately.

The equation for h(n) is calculated iteratively, looping through the values (v) of the tiles 1-15 as follows:

1. Find the index i in the board array of value v
2. Determine which row and column v is in for the state's board
    a. Row is determined by performing floor division to divide i by 4. This results in 0-3 i values in row 0, 4-7 i values in row 1, and so on to row 3.
    b. Column is determined by finding the modulus of i divided by 4. This places i values { 0, 4, 8, 12 } in column 0, i values { 1, 5, 9, 13 } in column 1, and so on to column 3
3. Repeat steps 1 and 2 to determine which row and column v is in for the end board
4. Calculate the distance from state board to end board using the following equation:

$$distance = |endrow - boardrow| + |endcolumn - boardcolumn|$$

5. Determine the cost of moving tile v to its end spot using the following equation:

$$tile\ cost = distance * v$$

6. Add the tile cost to the total estimated cost and repeat loop to the next v

This process results in a cost estimate which accounts for the cost of moving every misplaced tile to its location in the end board. By calculating the distance in this way, the formula determines the distance from the current board to the end board for each tile. Calculating the board this way allows for the pieces to be moved different ways to reach their final destination along an optimal path.

In my use of this heuristic algorithm, I have found that the estimated cost is never more than what the actual cost is. This means the algorithm is admissible. I must note that I have never run a testcase in which the algorithm underestimates the cost, but I believe that is okay. In comparing the heuristic algorithms that I submitted for Assignment 1 and my new heuristic algorithm, I have found that the number of moves for the optimal path is the same for each test case. However, the algorithm described in this document generates fewer open nodes than the old algorithms. Similarly, each of the test cases provided shows that fewer nodes were added to the closed list as well. This means that while the number of moves are the same between the algorithms, the algorithm described in this document is optimal because it opens fewer nodes when finding the optimal path. Additionally, I found that the new algorithm was able to solve a start layout of [7 6 5 4 3 2 1 0 9 10 11 12 13 14 15] to [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15], reversing the order of the first 8 tile spaces. I found that previous algorithms I tried crashed before this problem could be solved, but this algorithm was able to find a solution. Because of this and the enhanced performance of the assigned test cases, I argue that this algorithm is an optimal and admissible heuristic algorithm.