CS561A1Written

1. I used the heuristic given in the notes that counts the number of misplaced tiles. This heuristic is admissible because at most it can only be 8. Thus, it never exceeds the number of moves needed to reach the goal state and is therefore admissible.

2. If you look at the board and calculate the number of inversions, one can form a logical statement to test for solvability of a puzzle. An inversion would be a tile that comes before tiles that are smaller than it. For example 8 being in the top left of the 8-puzzle would give you 7 inversions. We can formulate that a puzzle with an odd width is solvable if the number of inversions is even.

To see that the proposed method truly shows solvability of a puzzle, we can observe how the number of inversions works on an 8-puzzle. No matter where a blank tile is moved the number of inversions either stays odd or even. If one moves a tile left or moves a tile right it does not change the odd or eveness of the inversions. If the tile is moved up, it passes by an even amount of more tiles so the number of inversions changes evenly.

Thus before starting my search I would count the number of inversions on the board by searching through the board and checking to see if the selected value is greater than the value to the right of it. After the number of inversions is found we can come up with a check that would look something like this,

```
if((node.boardWidth % 2 == 1) && (numInversions % 2 == 0))
    Puzzle is Solvable.
```

This check would be placed after reading in the initial puzzle board. If the statement is true then we can continue with the A* algorithm, otherwise the puzzle is marked as unsolvable.

3. I first tested my code by starting with a simple puzzle whose solution I knew and moving the blank tile one space. This was done so that I knew the minimum solution and could check the output of my program. I had issues with “hard” puzzles, and had to step through the code to make sure that the correct function values where being set. Of course, I was unable to step through the entire algorithm to completion. However, I did step through the first 8 complete iterations to make sure that my values in open and closed were being set correctly, and that the children of a selected node were being expanded and dealt with adequatly.