Tanner Heise 001223743
Zachary Kawa 001225030

For this project's implementation we used two classes, Board_Tile and Sliding_Solver. Each of the functions and data members that were outlined in the project description were implemented as specified. In addition to those functions and data members we added a handful of our own to both classes. They are as follows:

Board_Tile Functions	**Board_Tile Data Members**
Board_Tile(const string&, string): A constructor to build the next config *O(1)*	**String configFluid:** a changeable configuration member
getConfig(): A simple way to acquire a board's current configuration *O(1)*	**Int manhattanDistance:** easy place to store the ManhattanDistance for a BoardTile
print(): Print's a Board_Tile's configuration *O(1)*	**String endConfig:** an easy place to store the desired ending configuration of a board
findBlank(): find's the position of the "empty" tile in a Board_Tile *O(1)*	
findBlankFluid(): same as the last, just for a different data member *O(1)*	
move<*direction*>(): changes the configuration by moving 0 in the specified direction *iii*	
getManhattanDistance(): simple getter for Manhattan Distance *O(1)*	
operator<: compares two Board_Tiles with by their Manhattan Distance *O(1)*	

Sliding_Solver Functions	**Sliding_Solver Data Members**
makeHeap(vector<Board_Tile>): takes the list from nextConfigs and adds it to the minHeap (tileQueue) for future use *O(1)*	**String sc:** a place to store the starting configuration
	String ec: a place to store the ending configuration

Manhattan_Distance: O(1) as it has a maximum, and non variable, amount of operations it carries out.
nextConfigs: O(1) as it too at most carries out a set number of operations.
numMoves: O(1) this function is basically just a getter in the way it works, simply retrieves a value and returns it.

Solve_Puzzle: worst case running time of Solve_Puzzle is O(n) where n is the number of edges to potential configurations throughout the puzzle's solution.

We feel as if this could have been done quite a bit easier and probably better if the tileQueue stored the Manhattan distance of each next move rather than the entire Board_Tile object as that would make the comparison of each move's value much quicker and easier. However we're sure certain other problems would arise that would make it just as difficult and complex as the implementation that we were required to use for the project as it is.