Umar Sultan Sudoku Written Work

* Task 9

The function entriesToDel is supposed to generate n number of random coordinates in a 4 by 4 grid. The user passes the number of coordinates with n. The problem with the function is that it does not always generate n number of unique coordinates, there are duplicates sometimes which results in there being less blank spaces than specified by the user.

One possible solution to the issue would be to compare our newly generated coordinates against what has already been pushed to the array each iteration. We only push the new coordinates if no duplicate was found in the existing array the new coordinates are being pushed to. You could use linear search to compare these values if you convert them all to string and back, but you would need to use to separate arrays to implement such a method.

* Task 10

There are some limitations to this implementation of the sudoku algorithm. For starters, there is no check on the parameters being passed to functions to make sure if they are formatted correctly for the function to use. One such example would be row that we pass to makeRows, you can pass any number of elements in the row array that would crash our algorithm since its built with a 4 by 4 grid in mind. A simple if statement in the makeRows function could prevent an array that is smaller or larger than 4 elements from being passed. Data must be sorted automatically before it is used to ensure you don't run into any unforeseen errors.

There is also error checks built in elsewhere, there are plenty of functions that could run into an error because of a wrong data type or size being passed. Error messages that tell you what the error is would be helpful.

This is also a very resource intensive algorithm. It relies on linear search for most of its sorting and that linear search must be called multiple times to sort a single array. This repeated use of a brute search algorithm on a single array uses valuable time and increases the overhead the cpu has to handle. A possible solution would be to use a different search algorithm to sort through data such as binary search which would cut back on resources if data is already sorted and checked when it is passed. Also not needing to call a search algorithm multiple times for a single array would also help reduce resource usage.

This algorithm is built with a 4 by 4 puzzle in mind and is hard coded to suit this format. A more dynamic approach where it can handle any even input that can be broken into an even number of sub grids would making this algorithm more versatile and adaptable. This could be made more modular by breaking the functions up into different files so it can be worked on individually.