Solve() Method

- This approach is roughly based upon how back tracking can be used to solve a Sudoku puzzle which like our problem should have unique elements per row and column.
- Using Sudoku approach, I have tried to minimize the number of traversals needed to verify if a solution is possible of not.
- In the beginning, we know that all cell groups of "=" operations will only take the output value as such groups only have one cell and no other value is possible. So, I am directly placing those values in their respective cells by directly accessing those cells using index value rather than traversing through whole puzzle matrix because these operations doesn't need any traversal to determine if a value satisfies the operation condition or not.
- After that, I am removing the groups of "=" groups from the traversal list as these value will not be effected during further process. Doing so, it eliminates all groups of "=" operations, which creates minimized traversal list of those cells which are yet to be filled, eliminates unnecessary looping.
- For other group operations (+,-,/,*), I have created a method called generateMatrix() which will return a final solved puzzle matrix if any solution is possible otherwise returns false.
- generateMatrix() is called after assigning values for each group cell of every "=" operation.
- It takes **operation List**, which contains remaining operations along with their index values, array of each operation's **group name** and finally index value which denotes first index of the group name array as initial parameter.
- Basically, generateMatrix() checks every possible permutations of list of all possible value sets for each group with each possible value set of other operations group until a solution is found or no solution is possible.
- We start with group at first index in the group_name[]. We get list of every possible value set of this group along with index location of each cell in the group.
- Now, using looping traverse through each possible set and assign those values at each index location of that group's cell.

- While assigning, check if the assigning value is already present in either row or column. If it is, discard this value set, reset those index positions and choose the next value set in the possible value list.
- If the values are unique for each cell of group, assign them and call the generateMatrix() with updated index value as (index+1).
- Before calling generateMatrix(), check if the puzzle matrix is filled completely or not, to determine if a solution is reached or not. If it is, return to the solve() and return true indicating that a solution is possible.
- By recursively calling generateMatrix(), the above mentioned process will repeat for next group in the group_name[] until every element in the array is traversed and every possible combinations of possible sets for each group is tested.
- Using back tracking algorithm to create possible solution with help of generated possible value sets and index positions of each group cell, we can make process of finding solution matrix efficient by eliminating time consuming process of looping, traversing through whole puzzle matrix to check if a value is possible at current index by traversing to few number of indexes at a time for each group.