HW1 Report

Contributor(s): Luis Garcia

Please have pity on me since I tried to code this assignment alone. With that said, I did all the work on the assignments.

Part 1:

For the first backtracking implementation, which I named *BTstarter,* I found a way to extract the information from a .txt file using a scanner, which asks the user for the file name with the extension to run from anywhere using command Prompt. Then I built two double arrays, one containing int's and the other containing strings. I called the int array board and the string array charBoard, the former being used to store the values for the solution and the rather being used to identify the group at each position. Next, I used a bunch of HashMaps to store numVar (which keeps track of the number of variables in each group), groupOperand (used to extract operands: "/", "-", etc.), and three groupValues (two to modify, and one to keep track of the original value of each group).

Taking inspiration from a similar problem, the 4 Queens, I implemented backtracking with the method solve, which uses recursion to try every possible solution that meets the restrictions. These restrictions come in the form of helper methods called crossCheck (which checks if placing number based on horizontal and vertical restrictions) and equation (which is used as a portal to send it to other helper methods that checks whether or not it is a valid number). Within these helper methods the groupValues are altered. For subtraction and division, I found a pretty good way to limit the number of possible values. However, for addition and multiplication, I ended up just passing any valuable that is greater than 0 or that is a true integer, and checking whether the variables where valid while passing the last variable in that group. Because I modify the groupValues when passing numbers, I needed a helper method to reverse the effect when a value ended up not being right, which I called reverseEquation. Once it finishes the solve method, it prints the board and the number of nodes. Having that said, although it works fine, when given the sample problem, it iterates through 672 nodes, which is too much.

Part 2:

For my best backtracking implementation, which I named BestBackTracking, I started out the same way, but this time I implemented a node class called Square. My main goal for implementing the Square class was to compare the positions within each member of a group in order to pass more carefully thought out values. For this reason, there is a method called beside which compares two squares to check whether or not they are beside each other, or more accurately share the same x and y coordinates within a board. Moreover, I replaced the two original boards with one singular board that contains Square objects which contained all the information I wanted to have.

Next, like the BTstarter, I called a method called solve which uses recursion to iterate through all the possible values and placing the ones which pass the crossCheck and Equation methods. However, I implemented each equation method a bit differently. Each method requires a Square and the value that is currently being checked to see if it has an actual possibility of contributing to solving the equation at that given time. Because of this, it goes through each calling in the form of cases which is determined by the id of the square which is determined by the order in which it is read first when looping through the board array. Moreover, it returns true if it at least finds one possible solution given the constraints of the group and the board. Furthermore, it also checks whether the possible solution could be placed at that given time using crossCheck on each member as to not waste time on a solution that does not really work to begin with. As this implementation does not modify the groupValues, it does not reverse equation to reverse the effects of each iteration. In the end, given the sample problem, it iterates through 129 nodes which is close to 6 times the starter implementation.

Resources:
My inspiration came from: https://www.geeksforgeeks.org/n-queen-problem-backtracking-3/
And I copied and pasted a print method from StackOverflow and modified it in Part 2 to fit the Square board.