My heuristics in this lab consisted of tons of preprocessing and a backtracking algorithm with MRV heuristic. I first take the board input and make it a list of a list with the rows of the board making up the interior lists. Then, I create a dictionary of the ships (submarine, destroyer, cruiser and battleship) and fill the adjacent and diagonal squares near them with water. I consider the location of the ship hint given. For example, if the ship hint is in the top right corner square and it's a "R", I fill the square below the "R" and left diagonal to the "R" with water only and leave the square to the left of the "R" as 0. This is because we can't assume whether it'll be a destroyer or cruiser or battleship yet. After filling all the hint ships with water near them, I update the row and column integer lines that were given initially with the new row and column integers after considering the hints and their surrounding water. I can fill in water for every row and column that has 0 ships in it currently. This allows for a reduced domain of squares to consider when I am doing Backtracking with MRV. Since I perform the cell-based approach, I find the possible domains for every square that isn't currently water or a hint ship. So for every '0' square, I find its location on the board and check for potential ship parts it can be, based on filled squares adjacent to it. This method, although tedious, reduces my checking domain incredibly when I do backtracking. After performing these checks, I have a dictionary of indexes that are currently unassigned and the potential ship parts or water it can be assigned to in its domain. I then perform MRV with this domain, checking to see which domain is the smallest and then assigning those indexes and domains to the front of the dictionary. So essentially now I have a sorted dictionary of smallest domains to largest domains with their respective indices as the key. For the search, I use backtracking, which is kind of similar to depth first search in this case. I look through all the possibilities for my variables and domains that don't violate the three constraints I have in place: the row constraint with each row having a specific number of ships that it must have, the column constraint with each column having a specific number of ships that it must have, and finally the ship constraint which checks for the number of ships being placed (and this goes hand in hand with my domain checking preprocessing step which only allows valid ships to be placed in specified indices).