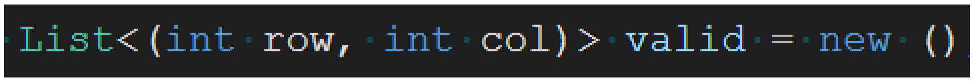
Eight queens code implementation

Variables:

1. Valid :

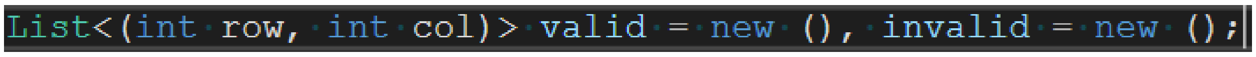
   list of row , column pairs are stored and are valid based on the conditions that they do not have the same columns. One of the valid solution is (0,0), (1, 4), (2, 7), (3, 5), (4, 2), (5, 6), (6, 1), (7, 3)



```
List<(int row, int col)> valid = new ()
```

2. Invalid :

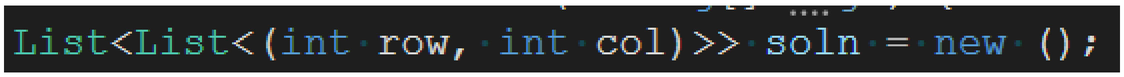
   list of row, column pairs which are invalid to place the next queen in the immediate columns. For (0,0) invalid places are (1, 0), (2, 0) till (7, 0) and the diagonals (1,1), (2,2), (3,3) till (7,7).



```
List<(int row, int col)> valid = new (), invalid = new ();
```

3. Soln :

   List of all possible valid places that can be obtained by placing the queen at each location and checking with the invalid places.

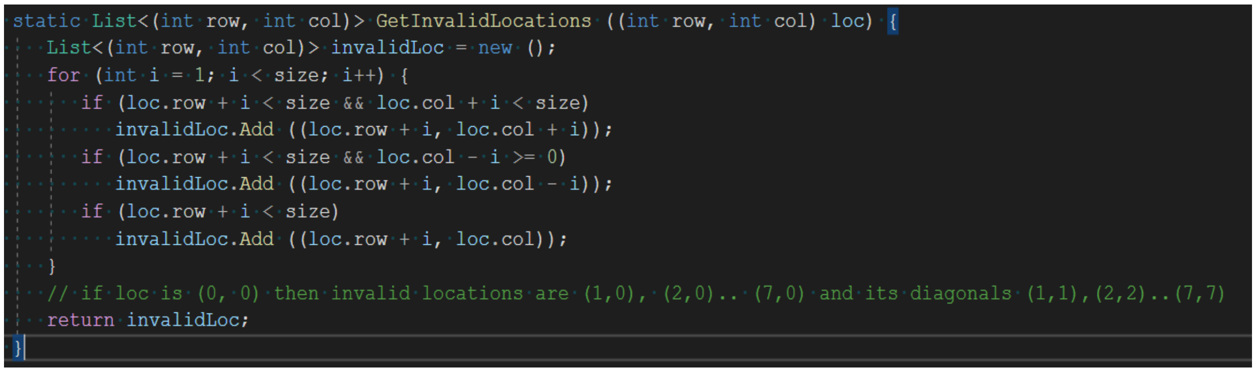


```
List<List<(int row, int col)>> soln = new ();
```

Methods:

1. GetInvalidLocations :

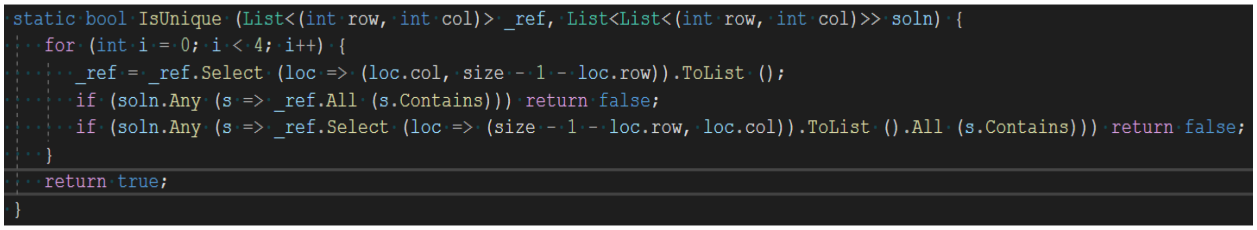
   Takes row - column pair as a parameter and returns all possible invalid locations for the corresponding pair. Example : if (0, 0) is given then it returns it iterates through row index till 7 and adds all invalid row - column pairs to the list. Invalid condition is that row - column pair should not have the input column that is 0 and should not be diagonal to the given input pair and hence (1, 0) (2, 0) till (7, 0) are invalid and also the diagonals (1,1) , (2,2) , (3,3) till (7, 7) are invalid. All these pairs are added to the list and returned.



```
static List<(int row, int col)> GetInvalidLocations ((int row, int col) loc) {
  List<(int row, int col)> invalidLoc = new ();
  for (int i = 1; i < size; i++) {
    if (loc.row + i < size && loc.col + i < size)
      invalidLoc.Add ((loc.row + i, loc.col + i));
    if (loc.row + i < size && loc.col - i >= 0)
      invalidLoc.Add ((loc.row + i, loc.col - i));
    if (loc.row + i < size)
      invalidLoc.Add ((loc.row + i, loc.col));
  }
  // if loc is (0, 0) then invalid locations are (1,0), (2,0).. (7,0) and its diagonals (1,1),(2,2)..(7,7)
  return invalidLoc;
}
```

2. IsUnique :

Takes all possible valid lists of row-column pairs and one of the row-column pairs as parameters and checks whether the duplicate of row - column pair exists in the list after rotating the row - column pair to 90, 180, 270 and mirroring them correspondingly. If duplicate exists in the list, returns false else returns true;

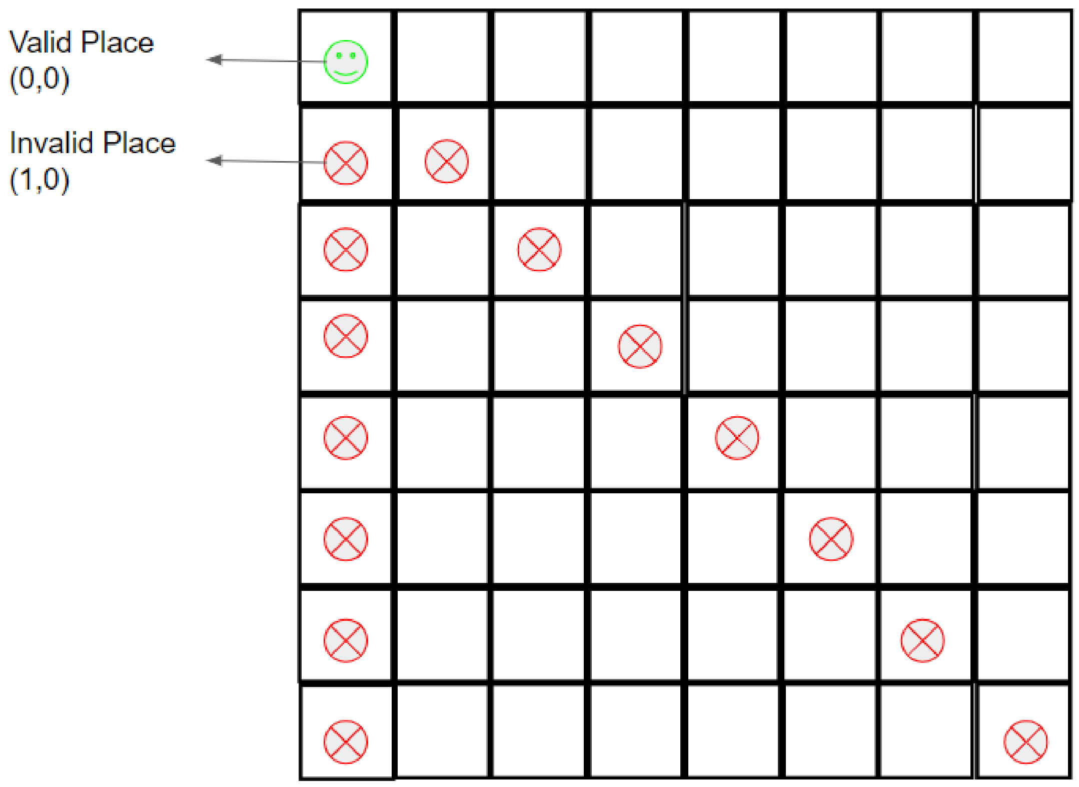


```
static bool IsUnique (List<(int row, int col)> _ref, List<List<(int row, int col)>> soln) {
  for (int i = 0; i < 4; i++) {
    _ref = _ref.Select (loc => (loc.col, size - 1 - loc.row)).ToList ();
    if (soln.Any (s => _ref.All (s.Contains))) return false;
    if (soln.Any (s => _ref.Select (loc => (size - 1 - loc.row, loc.col)).ToList ().All (s.Contains))) return false;
  }
  return true;
}
```

Logic:

1. Place a Queen at (0, 0)
2. Find all invalid row - column pairs for (0, 0)
3. Place the next queen at (1, 0) and check whether it exists in invalid pairs.
4. If it doesn't exist, add to the valid pairs else move to (1,1) till (1,7) and add valid pair
5. Iterate the for loop till (0, 7) and try to find all the valid pairs by calculating the invalid pairs and check with that correspondingly.
6. If valid pairs count becomes 8, then repeat the steps from 1 to 5 by incrementing the column by 1. I.e, place the queen at (0,1) and repeat the steps from 1 - 6
7. If the valid pairs count is less than 8 after iterating through all rows, then take the last pair from the valid pair and increment its column value by 1 and check for the invalid condition.

   Example : valid pairs have (0, 0) (1, 2) (2, 5) (3, 4) then take the last pair (3, 4) increment its column value by 1 i.e, (3,5) and check the invalid pairs, if it doesn't exist, then add to the valid pair and run iteration from 4th row till 7 and look for all possible pairs. If (3, 5) already exists in invalid pairs then increment the column value by1 till 7 and repeat finding the valid pairs. In this way back tracking is working to find all possible pairs.


Valid Place
(0,0)
Invalid Place
(1,0)