Sudoku Pair

Question 1

1. **Algorithm:-**
   We formed an encoding covering properties of the sudoku pair and found a solution of this encoding.
2. **Literals**
   For each position in both sudokus we have formed k*k literals, each depicting which value that cell holds.
   In implementation we have formed two 3D arrays a,b in which a[i][j][l] means i,j position of sudoku (starting position 0,0) holds value l+1. We have mapped these two arrays to natural numbers meaning their literal number.
3. **Clauses**
   Clauses are involved to satisfy the following constraints.
   a) Each position will hold only one value :- only one among k*k literals corresponding to a position can be true.
   b) Each row will hold only one value :- In a row of sudoku, only one literal corresponding to one value among [1,k*k].
   c) Each column will hold only one value :- In a block of sudoku, only one literal corresponding to one value among [1,k*k].
   d) Each block will hold only one value :- In a block of sudoku, only one literal corresponding to one value among [1,k*k].
   e) No two values of both sudoku at the same position are the same :- For the same position, for a value , at most one literal among both literals in sudoku can be true.
   Or ,

```
for all (i,j)
        For all (l)
                at most one is true among(a[i][j][l],b[i][j][l]).
```

   For implementing (a,b,c,d) we have used the equals function from the pysat.card library.
   This function is further broken in two functions atleast, atmost.
   Atleast is **or of all literals**
   Atmost is **and** of **or** of all **pairs of literals' negation.**

4. **Assumptions :-**
   All the values already given in both sudokus were passed as assumptions in the solver function.

Question 2

1. **Algorithm: -**

   Firstly we make a empty sudoku with each cell empty then filled a block along diagonal randomly with [1,k*k] ,we choose diagonal because it gives the most random sudoku at every attempt.
   Now we found out the solution of this sudoku using clauses ,literals ,and assumptions as given in Q1 .This will be the answer of the sudoku pair we are going to generate.
   Now we updated our assumption list with all the values given and added a clause which is or of all values in answer sudoku's negation ,this ensures we will never get the same sudoku pair again.
   Now we started emptying each cell in pairs and changed the assumptions accordingly and solved this using pysat.
   If we get a solution of this we put this cell again and change assumptions accordingly ,this ensures a unique solution of the sudoku pair.
   Thus we get a maximal and unique sudoku pair.
2. **Literals**

   For each position in both sudokus we have formed k*k literals, each depicting which value that cell holds.
   In implementation we have formed two 3D arrays a,b in which a[i][j][l] means i,j position of sudoku (starting position 0,0) holds value l+1. We have mapped these two arrays to natural numbers meaning their literal number.
3. **Clauses**

   Clauses are involved to satisfy the following constraints.
   a) Each position will hold only one value :- only one among k*k literals corresponding to a position can be true.
   b) Each row will hold only one value :- In a row of sudoku, only one literal corresponding to one value among [1,k*k].
   c) Each column will hold only one value :- In a block of sudoku, only one literal corresponding to one value among [1,k*k].
   d) Each block will hold only one value :- In a block of sudoku, only one literal corresponding to one value among [1,k*k].
   e) No two values of both sudoku at the same position are the same :- For the same position, for a value , at most one literal among both literals in sudoku can be true.
   Or ,
   ```
   for all (i,j)
           For all (l)
   ```

at most one is true among(a[i][j][l],b[i][j][l]).

f) After getting the solution sudoku ,we add one more clause which ensures this solution will not come again.

For implementing (a,b,c,d) we have used the equals function from the pysat.card library.

This function is further broken in two functions atleast, atmost.

Atleast is **or of all literals**

Atmost is **and** of **or** of all **pairs of literals' negation.**

4. Assumptions :-

All the values already given in both sudokus were passed as assumptions in the solver function.