Combinatorial Alphabet Problem :Design a game considering following points

- Objective is to fill a 9×9 grid with alphabets.
- Each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the alphabets from A to I.
- Entries in 9×9 grid not more than 18.

Solution :

Main Logic

The main target of the problem is to have unique characters in all cells of the specified blocks, rows and columns of the 9x9 grid.

Modules Used :

1. **CLIPFD**

First of all to accept the Input I use the sudoku function. The sudoku is solved using this function. The input is accepted as a 2 Dimensional List. The input is in such a way that all the the empty cells are filled with 'X' and all the other cells are filled with respective alphabets between 'A' and 'I' .

A sample input is shown below:

```
sudoku( [
        ['B','X','X','X','I','X','X','X','A'],
        ['C','X','I','X','X','G','X','X','X'],
        ['X','X','A','X','D','X','X','G','X'],
        ['X','F','X','X','X','X','X','X','X'],
        ['X','X','X','X','X','C','X','X','X'],
        ['X','X','H','F','X','X','G','I','X'],
        ['F','X','X','G','X','X','H','X','X'],
        ['A','B','C','X','X','H','X','X','X'],
        ['X','H','G','X','X','D','C','X','X']
    ]).
```

Now I am using a function is isAlpha()/2 which accepts two arguments. It converts the Input which I gave to Integers between 1 to 9. As IsAlpha() is specified for digits from 1 to 9. And for the blank space a we don't assign any value.

```
isAlpha('A',B):-
    B is 1.
isAlpha('B',B):-
    B is 2.
isAlpha('C',B):-
    B is 3.
isAlpha('D',B):-
    B is 4.
isAlpha('E',B):-
    B is 5.
isAlpha('F',B):-
    B is 6.
isAlpha('G',B):-
    B is 7.
isAlpha('H',B):-
    B is 8.
isAlpha('I',B):-
    B is 9.
isAlpha('X',_).
```

Using isAlpha() I convert all the Alphabets to integers from 1 to 9 and store the resulting 2 Dimensional list in another Variable. Then using ins/2 functionality in the CLIPFD Library we assign values to all the blank positions in the 2 Dimensional list. The range of values are between 1 to 9.

```
Vars =      [A1,B1,C1,D1,E1,F1,G1,H1,I1,
             A2,B2,C2,D2,E2,F2,G2,H2,I2,
             A3,B3,C3,D3,E3,F3,G3,H3,I3,
             A4,B4,C4,D4,E4,F4,G4,H4,I4,
             A5,B5,C5,D5,E5,F5,G5,H5,I5,
             A6,B6,C6,D6,E6,F6,G6,H6,I6,
             A7,B7,C7,D7,E7,F7,G7,H7,I7,
             A8,B8,C8,D8,E8,F8,G8,H8,I8,
             A9,B9,C9,D9,E9,F9,G9,H9,I9],
Vars ins 1..9,
```

Now we have to check all the rows, columns and the blocks mentioned in the Question have unique set of elements or not. For this we use the all_different()/2 functionality of CLIPFD Library. We accepts a list and will be true only if all the elements of the list are unique. We pass the corresponding variables of all the rows, columns and blocks mentioned in the question to the all_different() function to check whether it is unique or not and will produce a 2d list in such a way.

To check all rows

```
all_different([A1,B1,C1,D1,E1,F1,G1,H1,I1]),
all_different([A2,B2,C2,D2,E2,F2,G2,H2,I2]),
all_different([A3,B3,C3,D3,E3,F3,G3,H3,I3]),
all_different([A4,B4,C4,D4,E4,F4,G4,H4,I4]),
all_different([A5,B5,C5,D5,E5,F5,G5,H5,I5]),
all_different([A6,B6,C6,D6,E6,F6,G6,H6,I6]),
all_different([A7,B7,C7,D7,E7,F7,G7,H7,I7]),
all_different([A8,B8,C8,D8,E8,F8,G8,H8,I8]),
all_different([A9,B9,C9,D9,E9,F9,G9,H9,I9]),
```

To check all columns

```
all_different([A1,A2,A3,A4,A5,A6,A7,A8,A9]),
all_different([B1,B2,B3,B4,B5,B6,B7,B8,B9]),
all_different([C1,C2,C3,C4,C5,C6,C7,C8,C9]),
all_different([D1,D2,D3,D4,D5,D6,D7,D8,D9]),
all_different([E1,E2,E3,E4,E5,E6,E7,E8,E9]),
all_different([F1,F2,F3,F4,F5,F6,F7,F8,F9]),
all_different([G1,G2,G3,G4,G5,G6,G7,G8,G9]),
all_different([H1,H2,H3,H4,H5,H6,H7,H8,H9]),
all_different([I1,I2,I3,I4,I5,I6,I7,I8,I9]),
```

To check all blocks

```
all_different([A1,A2,A3,B1,B2,B3,C1,C2,C3]),
all_different([D1,D2,D3,E1,E2,E3,F1,F2,F3]),
all_different([G1,G2,G3,H1,H2,H3,I1,I2,I3]),

all_different([A4,A5,A6,B4,B5,B6,C4,C5,C6]),
all_different([D4,D5,D6,E4,E5,E6,F4,F5,F6]),
all_different([G4,G5,G6,H4,H5,H6,I4,I5,I6]),

all_different([A7,A8,A9,B7,B8,B9,C7,C8,C9]),
all_different([D7,D8,D9,E7,E8,E9,F7,F8,F9]),
all_different([G7,G8,G9,H7,H8,H9,I7,I8,I9]),
```

Now we have to convert all the numbers in the 2D list to the corresponding Alphabets. For that I am using isNumber()/2 method which has 2 arguments. We pass the corresponding variables of the 2D list which we obtained as the first argument to this function and obtain the Alphabet version through the second argument of the function. Access that using a new variable and store it in a new 2 Dimensional list.

```
isNumber(1,'A').
isNumber(2,'B').
isNumber(3,'C').
isNumber(4,'D').
isNumber(5,'E').
isNumber(6,'F').
isNumber(7,'G').
isNumber(8,'H').
isNumber(9,'I').
```

```
Ans = [VLA1,VLB1,VLC1,VLD1,VLE1,VLF1,VLG1,VLH1,VLI1,
              VLA2,VLB2,VLC2,VLD2,VLE2,VLF2,VLG2,VLH2,VLI2,
              VLA3,VLB3,VLC3,VLD3,VLE3,VLF3,VLG3,VLH3,VLI3,
              VLA4,VLB4,VLC4,VLD4,VLE4,VLF4,VLG4,VLH4,VLI4,
              VLA5,VLB5,VLC5,VLD5,VLE5,VLF5,VLG5,VLH5,VLI5,
              VLA6,VLB6,VLC6,VLD6,VLE6,VLF6,VLG6,VLH6,VLI6,
              VLA7,VLB7,VLC7,VLD7,VLE7,VLF7,VLG7,VLH7,VLI7,
              VLA8,VLB8,VLC8,VLD8,VLE8,VLF8,VLG8,VLH8,VLI8,
VLA9,VLB9,VLC9,VLD9,VLE9,VLF9,VLG9,VLH9,VLI9],
printSudoku(Ans).
```

Now list is printed using the printSudoku() function which I defined using format() function.

```
printSudoku([VLA1,VLB1,VLC1,VLD1,VLE1,VLF1,VLG1,VLH1,VLI1,
             VLA2,VLB2,VLC2,VLD2,VLE2,VLF2,VLG2,VLH2,VLI2,
             VLA3,VLB3,VLC3,VLD3,VLE3,VLF3,VLG3,VLH3,VLI3,
             VLA4,VLB4,VLC4,VLD4,VLE4,VLF4,VLG4,VLH4,VLI4,
             VLA5,VLB5,VLC5,VLD5,VLE5,VLF5,VLG5,VLH5,VLI5,
             VLA6,VLB6,VLC6,VLD6,VLE6,VLF6,VLG6,VLH6,VLI6,
             VLA7,VLB7,VLC7,VLD7,VLE7,VLF7,VLG7,VLH7,VLI7,
             VLA8,VLB8,VLC8,VLD8,VLE8,VLF8,VLG8,VLH8,VLI8,
VLA9,VLB9,VLC9,VLD9,VLE9,VLF9,VLG9,VLH9,VLI9]):-
format('The solved sudoku is :- '),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA1,VLB1,VLC1,VLD1,VLE1,VLF1,VLG1,VLH1,VLI1]),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA2,VLB2,VLC2,VLD2,VLE2,VLF2,VLG2,VLH2,VLI2]),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA3,VLB3,VLC3,VLD3,VLE3,VLF3,VLG3,VLH3,VLI3]),nl,
format('----------------------'),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA4,VLB4,VLC4,VLD4,VLE4,VLF4,VLG4,VLH4,VLI4]),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA5,VLB5,VLC5,VLD5,VLE5,VLF5,VLG5,VLH5,VLI5]),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA6,VLB6,VLC6,VLD6,VLE6,VLF6,VLG6,VLH6,VLI6]),nl,
format('----------------------'),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA7,VLB7,VLC7,VLD7,VLE7,VLF7,VLG7,VLH7,VLI7]),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA8,VLB8,VLC8,VLD8,VLE8,VLF8,VLG8,VLH8,VLI8]),nl,
format('~w ~w ~w | ~w ~w ~w | ~w ~w ~w',[VLA9,VLB9,VLC9,VLD9,VLE9,VLF9,VLG9,VLH9,VLI9]),nl.
```

Output

```
?- sudoku([
    ['B','X','X','X','I','X','X','X','A'],
    ['C','X','I','X','X','G','X','X','X'],
    ['X','X','A','X','D','X','X','G','X'],
    ['X','F','X','X','X','X','X','X','X'],
    ['X','X','X','X','X','C','X','X','X'],
    ['X','X','H','F','X','X','G','I','X'],
    ['F','X','X','G','X','X','H','X','X'],
    ['A','B','C','X','X','H','X','X','X'],
    ['X','H','G','X','X','D','C','X','X']
]).
The solved sudoku is :-
B G F | H I E | D C A
C D I | A B G | E F H
H E A | C D F | B G I
-----------------------
G F B | D H I | A E C
I A E | B G C | F H D
D C H | F E A | G I B
-----------------------
F I D | G C B | H A E
A B C | E F H | I D G
E H G | I A D | C B F
true
```