Ex 8. Implementation of knowledge representation schemes - use cases

Name – Harsh Prasad

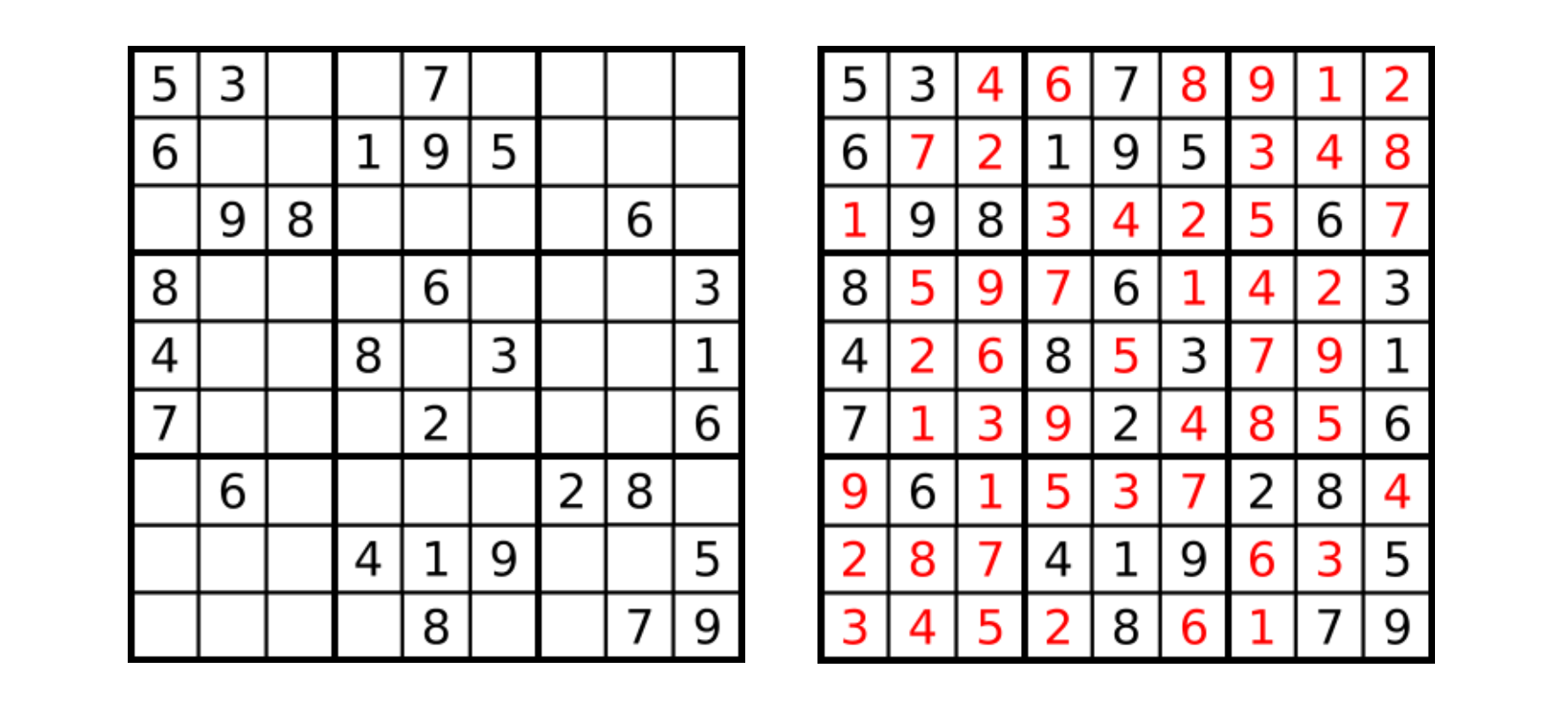
Reg no. – RA1811003010967

Date of exp. (presentation) – 05-04-2021

**Aim** : To solve Sudoku using Python language.

**Description**: Sudoku is a well-known puzzle game and popular for explaining search problems. Given an initial 9x9 grid of cells containing numbers between 1 and 9 or blanks, all blanks must be filled with numbers. You win Sudoku if you find all values such that every row, column, and 3x3 sub square contains the numbers 1–9, each with a single occurrence.

**Diagram :**



**Code :**

size = 9

#empty cells have value zero

matrix = [

[5,3,0,0,7,0,0,0,0],

[6,0,0,1,9,5,0,0,0],

[0,9,8,0,0,0,0,6,0],

[8,0,0,0,6,0,0,0,3],

[4,0,0,8,0,3,0,0,1],

[7,0,0,0,2,0,0,0,6],

[0,6,0,0,0,0,2,8,0],

[0,0,0,4,1,9,0,0,5],

[0,0,0,0,8,0,0,7,9]]

#print sudoku

def print\_sudoku():

for i in matrix:

print (i)

#assign cells and check

def number\_unassigned(row, col):

num\_unassign = 0

for i in range(0,size):

for j in range (0,size):

#cell is unassigned

if matrix[i][j] == 0:

row = i

col = j

num\_unassign = 1

a = [row, col, num\_unassign]

return a

a = [-1, -1, num\_unassign]

return a

#check validity of number

def is\_safe(n, r, c):

#checking in row

for i in range(0,size):

#there is a cell with same value

if matrix[r][i] == n:

return False

#checking in column

for i in range(0,size):

#there is a cell with same value

if matrix[i][c] == n:

return False

row\_start = (r//3)\*3

col\_start = (c//3)\*3;

#checking submatrix

for i in range(row\_start,row\_start+3):

for j in range(col\_start,col\_start+3):

if matrix[i][j]==n:

return False

return True

#check validity of number

def solve\_sudoku():

row = 0

col = 0

#if all cells are assigned then the sudoku is already solved

#pass by reference because number\_unassigned will change the values of row and col

a = number\_unassigned(row, col)

if a[2] == 0:

return True

row = a[0]

col = a[1]

#number between 1 to 9

for i in range(1,10):

#if we can assign i to the cell or not

#the cell is matrix[row][col]

if is\_safe(i, row, col):

matrix[row][col] = i

#backtracking

if solve\_sudoku():

return True

#f we can't proceed with this solution

#reassign the cell

matrix[row][col]=0

return False

if solve\_sudoku():

print\_sudoku()

else:

print("No solution")

**Output** :

A picture containing text, screenshot, computer, monitor

Description automatically generated

**Result :** The given sudoku problem is solved using python langauge