```
import re
import sys
import math

class InputSudokuToF2CSP:
    def __init__(self, inF = "", outF = "", matrix = [[]], n = 0) :
        self.inFile = inF
        self.outFile = outF

    def readFile(self) :
        self.inFile = input("Enter input file name:")
        self.outFile = input("Enter output file name:")

        file = open(self.inFile,"r")
        self.n = int(file.readline())

        self.matrix = [[0 for x in range(self.n)] for y in range(self.n)]


        numbersOnSlot = int(file.readline())

        while(numbersOnSlot > 0) :
            tuple = re.split("\s", file.readline())
            self.matrix[int(tuple[1])-1][int(tuple[0])-1] = int(tuple[2])
            numbersOnSlot -= 1

        file.close

    def writeFile(self) :
        f = open(self.outFile,"w+")

        f.write("Title: Sudoku"+str(self.n)+"x"+str(self.n)+"\n") #Titulo
        f.write("\n")
        f.write("Domains:\n1\nD1 "+"1.."+str(self.n)+"\n") #Dominios
        f.write("\n")
        f.write("Variables:\n"+str(self.n*self.n)+"\n") #Variaveis
        for i in range(1,self.n + 1) :
            for j in range(1,self.n + 1) :
                f.write("V"+str(i)+"-"+str(j)+" D1\n")

        f.write("\n")

        numberConstraints = 1
        stringTotal = ""

        for row in range(1,self.n + 1) :
            for col in range(1,self.n + 1) :
                constraintsROW = ""
                constraintsCOLlUMN = ""
                constraintsSQUARE = ""

                #Restricoes da Linha
                resColl = col+1
                for resCol in range(resColl,self.n + 1) :
                    constraintsROW = constraintsROW +
                    "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str(co
                    l)+"\n"+"V"+str(row)+"-"+str(resCol)+"\nReject:\n" + str(self.n)
                    +"\n"

                    for c in range(1,self.n+1) : #1..9 Rejects
                        constraintsROW = constraintsROW + str(c) + " " + str(c) + "\n"

                    numberConstraints += 1

                    constraintsROW = constraintsROW + "\n"

                stringTotal += constraintsROW

                #Restricoes da Coluna
                resRows = row+1
                for resRow in range(resRows,self.n + 1) :
                    constraintsCOLlUMN = constraintsCOLlUMN +
```

```
                    "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str(co
                    l)+"\n"+"V"+str(resRow)+"-"+str(col)+"\nReject:\n" + str(self.n)
                    +"\n"

                    for r in range(1,self.n+1) :
                        constraintsCOLlUMN = constraintsCOLlUMN + str(r) + " " +
                        str(r) + "\n"

                    numberConstraints += 1

                    constraintsCOLlUMN = constraintsCOLlUMN + "\n"

                stringTotal += constraintsCOLlUMN

                #Restricoes do Quadrado
                frontRows = row

                while((frontRows % math.sqrt(self.n)) != 0 ) :
                    frontRows += 1
                    frontCols = col
                    backCols = col

                    while(backCols %  math.sqrt(self.n) != 1) : #numero de Colunas
                    entre 0 e o numero
                        backCols -= 1
                        constraintsSQUARE = constraintsSQUARE +
                        "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+st
                        r(col)+"\n"+"V"+str(frontRows)+"-"+str(backCols)+"\nReject:\n"
                         + str(self.n) +"\n"

                        for b in range(1,self.n+1) :
                            constraintsSQUARE = constraintsSQUARE + str(b) + " " +
                            str(b) + "\n"

                        numberConstraints += 1
                        constraintsSQUARE = constraintsSQUARE + "\n"

                    while(frontCols %  math.sqrt(self.n) != 0) : #numero de Colunas
                    entre o numero e n
                        frontCols += 1
                        constraintsSQUARE = constraintsSQUARE +
                        "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+st
                        r(col)+"\n"+"V"+str(frontRows)+"-"+str(frontCols)+"\nReject:\n
                        " + str(self.n) +"\n"

                        for fr in range(1,self.n+1) :
                            constraintsSQUARE = constraintsSQUARE + str(fr) + " " +
                            str(fr) +"\n"

                        numberConstraints += 1
                        constraintsSQUARE = constraintsSQUARE + "\n"

                stringTotal += constraintsSQUARE

        acceptSQUARE = ""
        for row in range(1,self.n + 1) :
            for col in range(1,self.n + 1) :
                if self.matrix[row-1][col-1] != 0 :
                    acceptSQUARE = acceptSQUARE +
                    "C"+str(numberConstraints)+":\nVars:\n1\nV"+str(row)+"-"+str(col)+
                    "\nAccept:\n1\n"+str(self.matrix[row-1][col-1])+"\n\n"
                    numberConstraints += 1

        stringTotal += acceptSQUARE

        f.write("Constraints:\n"+str(numberConstraints-1)+"\n\n")
        f.write(stringTotal)
        f.write("Goal:\nSatisfy")

        f.close()
```

```
126   test = InputSudokuToF2CSP()
127   test.readFile()
128   test.writeFile()
129   print("DONE")
130
```