

```

1  import re
2  import sys
3  import math
4
5  class InputSudokuToF2CSP:
6      def __init__(self, inF = "", outF = "", matrix = [[]], n = 0) :
7          self.inFile = inF
8          self.outFile = outF
9
10     def readfile(self) :
11         self.inFile = input("Enter input file name:")
12         self.outFile = input("Enter output file name:")
13
14         file = open(self.inFile,"r")
15         self.n = int(file.readline())
16
17         self.matrix = [[0 for x in range(self.n)] for y in range(self.n)]
18
19
20         numbersOnSlot = int(file.readline())
21
22         while(numbersOnSlot > 0) :
23             tuple = re.split("\s", file.readline())
24             self.matrix[int(tuple[1])-1][int(tuple[0])-1] = int(tuple[2])
25             numbersOnSlot -= 1
26
27         file.close
28
29     def writeFile(self) :
30         f = open(self.outFile,"w+")
31
32         f.write("Title: Sudoku"+str(self.n)+"x"+str(self.n)+"\n") #Titulo
33         f.write("\n")
34         f.write("Domains:\n1\nD1 "+"1.."+str(self.n)+"\n") #Dominios
35         f.write("\n")
36         f.write("Variables:\n"+str(self.n*self.n)+"\n") #Variaveis
37         for i in range(1,self.n + 1) :
38             for j in range(1,self.n + 1) :
39                 f.write("V"+str(i)+"-"+str(j)+" D1\n")
40
41         f.write("\n")
42
43         numberConstraints = 1
44         stringTotal = ""
45
46         for row in range(1,self.n + 1) :
47             for col in range(1,self.n + 1) :
48                 constraintsROW = ""
49                 constraintsCOLUMN = ""
50                 constraintsSQUARE = ""
51
52                 #Restricoes da Linha
53                 resColl = col+1
54                 for resCol in range(resColl,self.n + 1) :
55                     constraintsROW = constraintsROW +
56                     "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str(co
57                     l)+"\n"+"V"+str(row)+"-"+str(resCol)+"\nReject:\n" + str(self.n)
58                     +"\n"
59
60                     for c in range(1,self.n+1) : #1..9 Rejects
61                         constraintsROW = constraintsROW + str(c) + " " + str(c) + "\n"
62
63                     numberConstraints += 1
64
65                     constraintsROW = constraintsROW + "\n"
66
67                 stringTotal += constraintsROW
68
69                 #Restricoes da Coluna
70                 resRows = row+1
71                 for resRow in range(resRows,self.n + 1) :
72                     constraintsCOLUMN = constraintsCOLUMN +

```

```

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

70
71         for r in range(1,self.n+1) :
72             constraintsCOLUMN = constraintsCOLUMN + str(r) + " " +
                str(r) + "\n"

73
74             numberConstraints += 1
75
76             constraintsCOLUMN = constraintsCOLUMN + "\n"
77
78             stringTotal += constraintsCOLUMN
79
80             #Restricoes do Quadrado
81             frontRows = row
82
83             while((frontRows % math.sqrt(self.n)) != 0 ) :
84                 frontRows += 1
85                 frontCols = col
86                 backCols = col
87
88                 while(backCols % math.sqrt(self.n) != 1) : #numero de Colunas
89                     entre 0 e o numero
90                     backCols -= 1
91                     constraintsSQUARE = constraintsSQUARE +
92                     "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str
93                     r(col)+"\n"+"V"+str(frontRows)+"-"+str(backCols)+"\nReject:\n"
94                     + str(self.n) +"\n"
95
96                     for b in range(1,self.n+1) :
97                         constraintsSQUARE = constraintsSQUARE + str(b) + " " +
98                         str(b) + "\n"
99
100                     numberConstraints += 1
101                     constraintsSQUARE = constraintsSQUARE + "\n"
102
103                     while(frontCols % math.sqrt(self.n) != 0) : #numero de Colunas
104                         entre o numero e n
105                         frontCols += 1
106                         constraintsSQUARE = constraintsSQUARE +
107                         "C"+str(numberConstraints)+":\nVars:\n2\n"+"V"+str(row)+"-"+str
108                         r(col)+"\n"+"V"+str(frontRows)+"-"+str(frontCols)+"\nReject:\n"
109                         + str(self.n) +"\n"
110
111                         for fr in range(1,self.n+1) :
112                             constraintsSQUARE = constraintsSQUARE + str(fr) + " " +
113                             str(fr) + "\n"
114
115                             numberConstraints += 1
116                             constraintsSQUARE = constraintsSQUARE + "\n"
117
118                     stringTotal += constraintsSQUARE
119
120             acceptSQUARE = ""
121             for row in range(1,self.n + 1) :
122                 for col in range(1,self.n + 1) :
123                     if self.matrix[row-1][col-1] != 0 :
124                         acceptSQUARE = acceptSQUARE +
125                         "C"+str(numberConstraints)+":\nVars:\n1\nV"+str(row)+"-"+str(col)+
126                         "\nAccept:\n1\n"+str(self.matrix[row-1][col-1])+"\n\n"
127                         numberConstraints += 1
128
129             stringTotal += acceptSQUARE
130
131             f.write("Constraints:\n"+str(numberConstraints-1)+"\n\n")
132             f.write(stringTotal)
133             f.write("Goal:\nSatisfy")
134
135             f.close()

```

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