Nathan Sucher

Homework 2

```
1a)
# SINCE + JULIUS = CAESAR

import constraint
import time


def equation(s, j, c, i, n, e, u, l, a, r):
    if (s*10000 + i*1000 + n*100 + c*10 + e) + (j*100000 + u*10000 + l*1000 + i*100 +
u*10 + s) == (c*100000 + a*10000 + e*1000 + s*100 + a*10 + r):
        return True


if __name__ == "__main__":
    start = time.time()

    problem = constraint.Problem()

    problem.addVariables("SJC", range(1, 10))
    problem.addVariables("INEULAR", range(10))

    problem.addConstraint(equation, "SJCINEULAR")
    problem.addConstraint(constraint.AllDifferentConstraint())

    solutions = problem.getSolutions()

    stop = time.time()
    print("Program took " + "{:.2f}".format(stop - start) + " seconds to find
solution(s).")

    print("\nNumber of Solutions: ", len(solutions))

    for solution in solutions:
        print("\nS: ", solution['S'])
        print("\nJ: ", solution['J'])
        print("\nC: ", solution['C'])
        print("\nI: ", solution['I'])
        print("\nN: ", solution['N'])
        print("\nE: ", solution['E'])
        print("\nU: ", solution['U'])
        print("\nL: ", solution['L'])
        print("\nA: ", solution['A'])
        print("\nR: ", solution['R'])
```

Solution:

Program took 77.75 seconds to find solution(s).

Number of Solutions: 1

S: 9

J: 7

C: 8

I: 2

N: 6

E: 5

U: 1

L: 3

A: 0

R: 4

```
1b)
# CHECK + THE = TIRES

import constraint
import time


def equation(c, t, h, e, k, i, r, s):
    if (c*10000 + h*1000 + e*100 + c*10 + k) + (t*100 + h*10 + e) == (t*10000 + i*1000 +
r*100 + e*10 + s):
        return True


if __name__ == "__main__":
    start = time.time()

    problem = constraint.Problem()

    problem.addVariables("CT", range(1, 10))
    problem.addVariables("HEKIRS", range(10))

    problem.addConstraint(equation, "CTHEKIRS")
    problem.addConstraint(constraint.AllDifferentConstraint())

    solutions = problem.getSolutions()

    stop = time.time()
    print("Program took " + "{:.2f}".format(stop - start) + " seconds to find
solution(s).")

    print("\nNumber of Solutions: ", len(solutions))

    for solution in solutions:
        print("\nC: ", solution['C'])
        print("\nT: ", solution['T'])
        print("\nH: ", solution['H'])
        print("\nE: ", solution['E'])
        print("\nK: ", solution['K'])
        print("\nI: ", solution['I'])
        print("\nR: ", solution['R'])
        print("\nS: ", solution['S'])
```

Solution:

Program took 13.37 seconds to find solution(s).

Number of Solutions: 1

C: 5

T: 6

H: 9

E: 4

K: 3

I: 0

R: 1

S: 7

```
1c)
# DO + YOU + FEEL = LUCKY

import constraint
import time


def equation(d, y, f, l, o, u, e, c, k):
    if (d*10 + o) + (y*100 + o*10 + u) + (f*1000 + e*100 + e*10 + l) == (l*10000 + u*1000
+ c*100 + k*10 + y):
        return True


if __name__ == "__main__":
    start = time.time()

    problem = constraint.Problem()

    problem.addVariables("DYFL", range(1, 10))
    problem.addVariables("OUECK", range(10))

    problem.addConstraint(equation, "DYFLOUECK")
    problem.addConstraint(constraint.AllDifferentConstraint())

    solutions = problem.getSolutions()

    stop = time.time()
    print("Program took " + "{:.2f}".format(stop - start) + " seconds to find
solution(s).")

    print("\nNumber of Solutions: ", len(solutions))

    for solution in solutions:
        print("\nD: ", solution['D'])
        print("\nY: ", solution['Y'])
        print("\nF: ", solution['F'])
        print("\nL: ", solution['L'])
        print("\nO: ", solution['O'])
        print("\nU: ", solution['U'])
        print("\nE: ", solution['E'])
        print("\nC: ", solution['C'])
        print("\nK: ", solution['K'])
```

Solution:

Program took 29.47 seconds to find solution(s).

Number of Solutions: 1

D: 5

Y: 8

F: 9

L: 1

O: 7

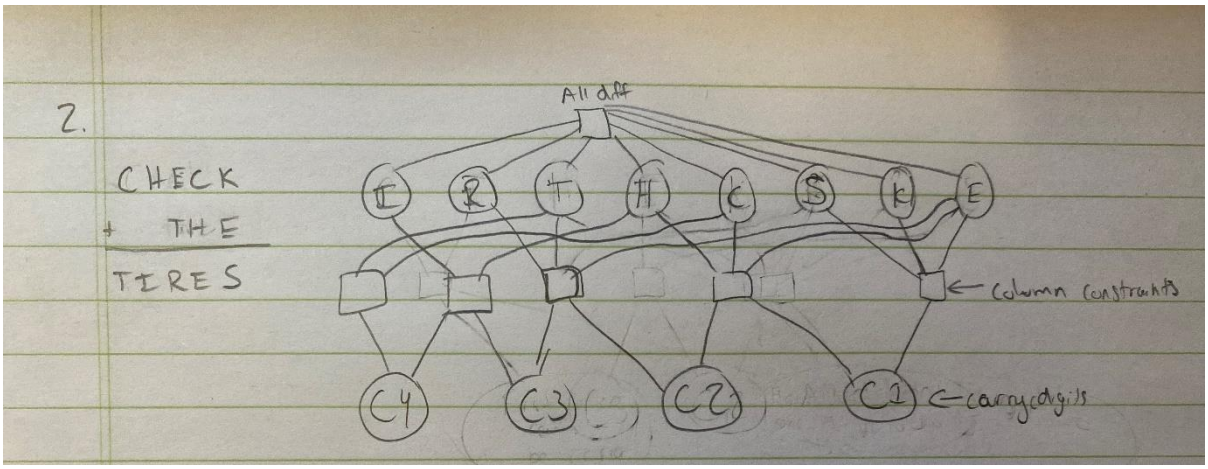
U: 0

E: 4

C: 3

K: 6

2)



3)

Homework 2

3. a) $(Food \wedge Drinks \rightarrow Party) \vee (Drinks \wedge Dance \rightarrow Party)$

$[\neg(Food \wedge Drinks) \vee Party] \vee [\neg(Drinks \wedge Dance) \vee Party]$

$(\neg Food \vee \neg Drinks \vee Party) \vee (\neg Drinks \vee \neg Dance \vee Party)$

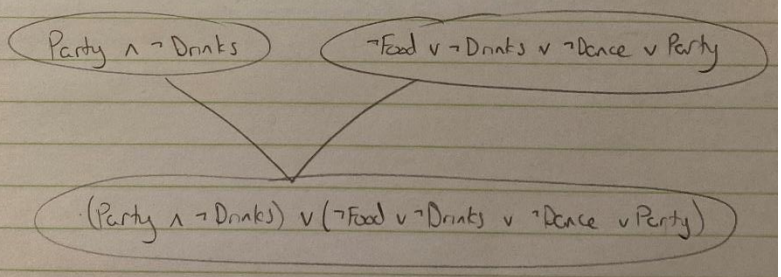
$\neg Food \vee \neg Drinks \vee \neg Dance \vee Party$

b) $Party \rightarrow Drinks$

$\neg Party \vee Drinks$

c)

Step	Formula	Derivation
1	$\neg Food \vee \neg Drinks \vee \neg Dance \vee Party$	Given
2	$Party \wedge \neg Drinks$	Negated conclusion
3		



At this stage, we decide to give up looking for a proof. We cannot find the proof since the goal is a singular 'and' clause. It does not mean the statement is false; we can just not prove by refutation.