Lab3 - Ex2

Varibile de decizie:

- x_1 = numarul de cadouri de tipul 1
- x_2 = numarul de cadouri de tipul 2
- x_3 = numarul de cadouri de tipul 3

Constrangeri:

1. Are nevoie de cel putin 100 de cadouri: $x_1 + x_2 + x_3 \geq 100$
2. Are nevoie de cel putin 5 cadouri de tipul 1 sau 2: $x_1 + x_2 \geq 5$
3. Are nevoie de cel putin 10 cadouri de tipul 3: $x_3 \geq 10$
4. Are la dispozitie 3 ore pentru cadouri: $x_1 + 2 * x_2 + 5 * x_3 \leq 180$
5. Are la dispozitie 300 de lei pentru cadouri: $3 * x_1 + 2 * x_2 + x_3 \leq 300$
6. Numarul de cadouri de fiecare fel nu poate sa fie negativ: $x_1 \geq 0; x_2 \geq 0; x_3 \geq 0$
7. Mos craciun vrea sa faca doar doua tipuri de cadouri: $x_1 == 0 \bigvee x_2 == 0 \bigvee x_3 == 0$

Vrem sa minimizam costul: $3 * x_1 + 2 * x_2 + x3$

Python script

```
from z3 import *

# Variabile
x1 = Int('x1')
x2 = Int('x2')
x3 = Int('x3')

# Constrangeri
#  1. Cel putin 100 cadouri
c1 = [x1 + x2 + x3 >= 100]

#  2. Cel putin 5 cadouri de tipul 1 sau 2
c2 = [x1 + x2 >= 5]

#  3. Cel putin 10 cadouri de tipul 3
c3 = [x3 >= 10]

#  4. Are la dispozitie 3 ore pentru cadouri
c4 = [x1 + 2*x2 + 5*x3 <= 180]

#  5. Are la dispozitie 300 lei pentru cadouri
c5 = [3*x1 + 2*x2 + x3 <= 300]

#  6. Numarul de cadouri nu poate sa fie negativ
c6 = [And(x1 >= 0, x2 >= 0, x3 >= 0)]

#  7. Doar doua tipuri de cadouri
```

```
c7 = [Or(x1 == 0, x2 == 0, x3 == 0)]

solver = Optimize()
solver.add(c1+c2+c3+c4+c5+c6+c7)
solver.minimize(3*x1 + 2*x2 + x3)

if solver.check() == sat:
    model = solver.model()
    print(model)
else:
    print("Problem is unsatisfiable")

print(solver.sexpr())
```

Rezultatul:

```
[x3 = 20, x2 = 0, x1 = 80]
```

= 20 de cadouri de tipul 3 si 80 de cadouri de tipul 1 cu un cost total de 260 RON

SMT file

```
(declare-fun x1 () Int)
(declare-fun x2 () Int)
(declare-fun x3 () Int)
(assert (>= (+ x1 x2 x3) 100))
(assert (>= (+ x1 x2) 5))
(assert (>= x3 10))
(assert (<= (+ x1 (* 2 x2) (* 5 x3)) 180))
(assert (<= (+ (* 3 x1) (* 2 x2) x3) 300))
(assert (and (>= x1 0) (>= x2 0) (>= x3 0)))
(assert (or (= x1 0) (= x2 0) (= x3 0)))
(minimize (+ (* 3 x1) (* 2 x2) x3))
(check-sat)
(get-model)
(get-objectives)
```