Exercício 1 (Investimento, Formulação, fácil)
Sejam u_i as unidades produzidas do produto $i \in P = [1,4]$, $x_i \in \mathbb{B}$ variáveis booleanas que indicam se o produto i é produzido, c_i e l_i o custo inicial e os lucros, respectivamente.

$$\begin{aligned}
\textbf{max} \quad & \sum_{i \in P} u_i l_i - \sum_{i \in P} c_i x_i & \\
\textbf{s.a} \quad & \sum_{i \in P} x_i \leq 2 & \text{No máximo dois tipos de produtos} \\
& x_3 \leq x_1 + x_2 & \text{Tipo 3 somente so tipo 1 ou 2} \\
& x_4 \leq x_1 + x_2 & \text{Tipo 4 somente so tipo 1 ou 2} \\
& u_i \leq 2000 \qquad \forall i \in P & \text{Limite produção (redundante)} \\
& u_i \leq 2000 x_i \qquad \forall i \in P & \text{Vínculo das variáveis}
\end{aligned}$$

Alternativas para a segunda restrição

$x_3 + x_4 \leq 2(x_1 + x_2)$	Mais fraco, a soma das duas restrições acima
$x_3 + x_4 \leq x_1 + x_2$	Mais forte, válido porque no máximo dois projetos

Exercício 2 (Sudoku)
O seguinte modelo em AMPL formaliza as restrições:

```
set digitos := 1 .. 9;
set linhas  := 1 .. 9;
set colunas := 1 .. 9;

var numero { linhas, colunas, digitos } binary;

maximize SomaDiagonalSuperior:
        sum { i in linhas, d in digitos } d*numero[i,i,d];

# cada quadro contém exatamente um digito
subject to QuadroMenor { i in linhas, j in colunas }:
        sum { d in digitos } numero[i,j,d] = 1;
# cada linha contém de cada dígito exatamente uma vez
subject to Linha { i in linhas, d in digitos }:
        sum { j in colunas } numero[i,j,d] = 1;
# cada coluna contém de cada dígito exatamente uma vez
subject to Coluna { j in colunas, d in digitos }:
        sum { i in linhas } numero[i,j,d] = 1;
# cada quadro maior contém cada dígito exatamente uma vez
subject to QuadroMaior { i in {1,4,7}, j in {1,4,7}, d in digitos }:
        sum { di in 0..2, dj in 0..2 } numero[i+di,j+dj,d] = 1;
```

Um exemplo de uma solucão é

8	3	1	4	2	5	8	9	6
4	8	2	6	9	1	3	5	7
5	6	9	3	7	8	2	1	4
6	2	5	7	1	3	9	4	8
1	9	3	2	8	4	6	7	5
8	4	7	5	6	9	1	2	3
9	5	4	8	3	2	7	6	1
3	1	6	9	5	7	4	8	2
2	7	8	1	4	6	4	3	9

Exercício 3 (Coloração de grafos)
Para um grafo $G = (V, A)$ com n vértices e $V = [n]$ seja $x_{ij} \in \mathbb{B}$ para $1 \leq v, j \leq n$ um indicador se o vértice v possui cor $i \in C$, com $C = [n]$ o conjunto de cores permitidos. (Permitimos até n cores, para garantir uma coloração.) Seja ainda $c_i \in \mathbb{B}$ para $i \in C$ uma variável auxiliar que indice se a cor i está usada.

minimiza	$\sum_{i \in C} c_i$		menor número de cores
sujeito a	$\sum_{1 \leq j \leq n} x_{vj} = 1$	$v \in V$	Garantir exatamente um cor por vértice
	$x_{ui} + x_{vi} \leq 1$	$i \in C, uv \in E$	Coloração viável
	$nc_i \geq \sum_{v \in V} x_{vi}$	$i \in C$	Define variáveis aux.

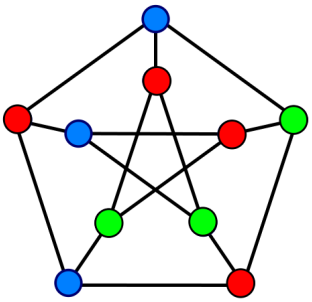
A implementação correspondente em AMPL para o caso do grafo de Peterson é

```
set V;                                  # set of vertices
set E within V cross V;                 # set of edges
var c { i in V } binary;                # color i set?
var x { i in V, j in V } binary;        # vertex i has color j?

minimize Colors:
        sum { i in V } c[i];

subject to OneColor { i in V }:
        sum { j in V } x[i,j] = 1;

subject to NoAdjacent { (i,j) in E, k in V }:
        x[i,k] + x[j,k] <= 1;

subject to DefineColor { i in V }:
        10*c[i] >= sum { j in V } x[j,i];
```

```
data;

set V := 1 2 3 4 5 6 7 8 9 10;
set E := (1,2) (1,3) (1,6) (2,7) (2,8) (3,4) (3,9) (6,5) (6,10) (4,5)
    (4,8) (5,7) (7,9) (8,10) (9,10) ;
# for use with cm
set A := (1,2) (1,3) (1,6) (2,7) (2,8) (3,4) (3,9) (5,6) (6,10) (4,5)
    (4,8) (5,7) (7,9) (8,10) (9,10) ;
```

end;

Solução:



Exercício 4 (Ponder this, Formulação, difícil)
Uma formulação em AMPL é

```
# cubes, numbers, and faces
set C := 1..25;
set F := 1..6;
set N := 0..31;

## set of different cubes
#set DC :=

# face variables
var x { C, F, N } binary;

# sharing variables
var y { C, C, N } binary;

# maximum number used
var M integer;

minimize maxnumber: M;

subject to exactlyonenumber { c0 in C, f0 in F }:
        sum { n0 in N } x[c0,f0,n0] = 1;

subject to noequalnumbers { c0 in C, f0 in F, f1 in F, n0 in N : f0 != f1
    }:
        x[c0,f0,n0]+x[c0,f1,n0] <= 1;

subject to definesharing1 { c0 in C, c1 in C, n0 in N : c0 != c1 }:
        y[c0,c1,n0] <= sum { f0 in F } x[c0,f0,n0];
subject to definesharing2 { c0 in C, c1 in C, n0 in N : c0 != c1 }:
        y[c0,c1,n0] <= sum { f0 in F } x[c1,f0,n0];
subject to definesharing3 { c0 in C, c1 in C, n0 in N : c0 != c1 }:
        y[c0,c1,n0] + 1 >= sum { f0 in F } ( x[c0,f0,n0] + x[c1,f0,n0]);

subject to shareonenumber { c0 in C, c1 in C : c0 != c1}:
        sum { n0 in N } y[c0,c1,n0] = 1;
```

```
subject to setmaxnumber { c0 in C, f0 in F, n0 in N }:
        M >= n0*x[c0,f0,n0];
```