NOPT042 Constraint programming: Tutorial 9 - Implicit constraints

In [1]:
```
%load_ext ipicat
```

```
Picat version 3.2#8
```

Example: Seesaw

Adam, Boris, and Cecil want to sit on a 10-feet long seesaw such that they are at least 2 feet apart and the seesaw is balanced. Adam weighs 36 lbs, Boris 32 lbs, and Cecil 16 lbs. Write a general model. You can assume that the length is even, the distance is integer, and that they can only sit at integer points.

(Problem from Marriott & Stuckey "Programming with Constraints", page 257. Instance from R. Barták's tutorial.)

In [2]:
```
!cat seesaw/instance1.pi
```

```
% sample instance
instance(NumPeople, Length, Distance, Weights) =>
    NumPeople = 3,
    Length = 10,
    Distance = 2,
    Weights = [36, 32, 16].
```

Possible decision variables?

- Position on the seesaw for each person.
- Distances between persons, position of the first person, and order of persons.
- Person or empty for each position on the seesaw.

Global constraints? Symmetry breaking? Multiple modeling? Search strategies?

In [3]:
```
!picat seesaw/seesaw.pi
```

```
*** Undefined procedure: main/0
```

Redundant (implicit) constraints

Redundant constraints do not restrict the solution set but rather express properties of a solution from a different viewpoint. This can lead to

- faster domain reduction,
- a significant boost in propagation,
- improved communication between variables.

We have already seen one example last week in the Magic sequence problem: adding the `scalar_product` constraint.

Example: Golomb's ruler

A Golomb's ruler is an imaginary ruler with n marks such that the distance between every two marks is different. Find the shortest possible ruler for a given n.

(The solution for N=28 was announced last week! The length is 585.)

- What length are you able to solve in reasonable time?
- Add suitable implicit constraints. (We will discuss this in class.)

In [4]:
```
!picat golomb/golomb.pi 4
```

```
CPU time 0.0 seconds. Backtracks: 49

length = 6
[0,1,4,6]
```

Homework: life

The goal is to find a still (stable) live organism within $N \times N$ subsquare of an (infinite) board of Conway's game of life. The organism must not change in time, the goal is to maximize its density (the number of live cells divided by N^2).

See the problem description on CSPLib.org. (But don't look at the solutions there. Also, in 2012 the problem was solved mathematically, but don't use the formula.)

Try to improve your model using symmetry breaking, implicit constraints, perhaps a good search strategy, etc.

```
picat life.pi 6
```

should return the optimal value of `18` and some representation of the organism.