MINIZINC HOMEWORK

For this homework assignment you will need the MiniZinc compiler and the Gecode constraint solver. The easiest option is to install the *Bundled binary package* from http://www.minizinc.org. The package contains the compiler, an IDE, and several solvers including Gecode. Everything should be up and running out of the box.

The homework consists of two problems described below. I have provided some sample inputs (`.dzn` files). Email both models (`.mzn` files) to bulin@karlin.mff.cuni.cz. If you are not sure how to express some constraints, check the examples from class, the tutorial, or talk to me.

The deadline is **Dec 7, 3:30pm**. Have fun!

Problem 1: Magic squares

A *magic square* is an $n \times n$ table filled with positive integers $1, 2, ..., n^2$ such that each cell contains a different integer and the sum of the integers in each row, each column, and both diagonals is equal. The sum is called the *magic constant*; n is the *order* of the magic square.

The input data consist of the order and a partially filled square. Write a MiniZinc model that will fill out the rest of the square (if it is possible) and compute the magic constant. Here is a sample input:

```
N = 3;
square =
[| 2, _, _
 | _, 5, _
 | _, 3, _
 |];
```

The output should be something like this:

```
The magic constant is 15.
The magic square:
2 7 6
9 5 1
4 3 8
----------
```

You can use the following code to get the output in a nice format (`magic_constant` is the name of the variable used to compute the magic constant):

```
output
["The magic constant is \(magic_constant).\n"] ++
["The magic square:\n"] ++
[ show_int(floor(log10(int2float(N*N))+1), square[i,j]) ++
  if j = N then "\n" else " " endif | i, j in 1..N];
```

Problem 2: Hungry hiker

A very hungry hiker is buying provisions for a backpacking trip. He is choosing from a list of food items. The supply is unlimited. Each item has a certain amount of calories (in kcal) and a certain weight (in grams). He can only carry food up to a given weight limit. What is the maximum amount of calories he can take?

Here is a sample input.

```
LIMIT = 6500;
ITEMS = {apple, beer, bread, carrot, pea, steak, water};
WEIGHT = [88, 564, 892, 415, 8, 410, 500];
KCAL = [40, 385, 615, 290, 1, 245, 5];
```

The optimal solution is 1 beer, 2 loaves of bread, and 10 carrots which sums up to 4515 kcal. Note that `ITEMS` is of type `enum`, declared by '`enum ITEMS;`'.