Solving Soft Constraint Problems in Autonomic Systems with MiniBrass (FAS* 2016)

Exercise 1 ***(MiniZinc: HelloWorld)***

Build a MiniZinc model `xopt.mzn` with a decision variable x taking values from 0 to 10, with constraints to ensures that x is divisible by 4, which outputs the value of x that gives the minimum value of $(x-7)^2$.
Test it using the precompiled IDE-bundle. Suppose you cannot use the `mod` function, how would you alternatively model that x is divisible by 4?

Exercise 2 ***(Arrays)***

Define a MiniZinc model `array.mzn` which takes an integer parameter n defining the length of an array of numbers x taking values from 0 to 9. Constrain the array so the sum of the numbers in the array is equal to the product of the numbers in the array. Output the resulting array. Test your model using the "all solutions" setting active in the IDE. Add a constraint to ensure that the numbers in the array are non-decreasing, i.e. $x[1] \leq x[2] \leq \ldots \leq x[n]$. This should reduce the number of similar solutions. How big a number can you solve with your model? Why do you think this happens?

Exercise 3 ***(Group Photo)***

Given a group of n people, we must arrange them for a photo. The best photo is when people are next to their friends, so the aim is to arrange them so that each person is next to (to the left or right) with as many friends as possible. The data for the problem is given as

```
n = <size of problem> ;
array[1..n,1..n] of var bool: friend;
```

where `friend[f1, f2]` means `f1` and `f2` are friends. You can assume that the friend array is symmetric. You should output a list of the people in their position to maximize the number of adjacent friends. For example given the data `groupphoto1.dzn`, you should output the placement of the guests as well as the objective value, i.e.,

```
Obj = 7; [4, 3, 5, 6, 8, 7, 1, 2]
```

Exercise 4 ***(Modeling Group Photo as a Soft Constraint Problem)***

Now we would like to refine our group photo model with constraint relationships. Start by examining the "pure" MiniZinc model `groupphoto-pure.mzn` and test it with `groupphoto1.dzn`. We will augment this model with preferences, starting with person 3 (Carla). She has three preferences

- `c1`: She would like to be placed next to person 2. (Hint: Use the provided `isNextTo` function.)
- `c2`: She would like to be placed in the second row.
- `c3`: Carla doesn't particularly like person 5. Hence, the Manhattan distance (provided as `manhattanDist`) between them should be greater than 4.

Constraint `c1` is most important to Carla, `c2` and `c3` are both less important than `c1` but incomparable. Write a preference model `groupphoto.mbr` that incorporates theses constraint relationships. Test the model (not in the IDE) using

```
mbr2mzn groupphoto.mbr
minisearch groupphoto.mzn groupphoto1.dzn
```

What is the best solution you get? What happens if we add another constraint `c4` that asks for person 5 not to be placed at either border (column 1 or m)?

Exercise 5 ***(Mentor Matching – Influence of Priorities)***

Inspect the pure MiniZinc model `student-company-matching.mzn` using the IDE. This core model only specifies feasible "matchings" but does not incorporate preferences. The MiniBrass file `student-company-matching.mbr` contains an empty declaration of two PVS, one for students, one for companies.
Add the following soft constraints for the students:

- Britney would like to work at Disney but even better, work at Warner
- Eminem wants to work at the University of Augsburg (of course) but this is less important than Britney being at Disney.

and for the companies:

- Disney would like that Christina works there.
- Disney would like that Falco works there.
- The university of Augsburg wants Britney.
- It is more important that Britney works at the University of Augsburg than Falco working at Disney.

Wrap both Constraint Relationship instances with a `ToWeighted` morphism and try both ways of doing a lexicographic product:

```
solve ToWeighted(students) lex ToWeighted(companies);
solve ToWeighted(companies) lex ToWeighted(students);
```

Finally test your approach with

```
mbr2mzn student-company-matching.mbr
minisearch student-company-matching-mbr.mzn
```