FIT5216: Modelling Discrete Optimization Problems
Inclass Task 18: Stable Goods Assignment

1 Problem Statement

The stable commodity assignment problem is defined as follows. We have a set of people, PERSON, and a set of goods, GOOD. Each person has a ranked list of which GOODs they like and how many would make them happy. We have a certain amount available of each good.

An correct assignment is that each person is given one type of the goods in their preference list, and they are given the number they want of that good.

A stable commodity assignment is a correct assignment, where there is no pair of people and goods such that each person would be happier with swapping the good with the other person, and we still have enough of each good to do the swap (since the two people may require different amounts of the good).

The data is given in a compact form.

```
enum PERSON;
enum GOOD;
array[GOOD] of int: available; % number of each good available
array[GOOD] of int: value;     % the value of each good unit
array[PERSON] of int: npref;   % number of preferences for each person
int: tpref = sum(npref);
set of int: PREF = 1..tpref;
array[PREF] of GOOD: good_pref; % good for each preference
array[PREF] of int: req_pref;   % number required for each preference
```

The decision variables are

```
int: maxreq = max(req_pref);
array[PERSON] of var GOOD: good;
array[PERSON] of var 0..maxreq: num;
```

The aim is to minimize the total value of goods given to the people while still being a stable assignment.

For example given data

```
PERSON = { A, B, C, D, E };
GOOD =      {   G1, G2, G3, G4 };
available = [   10, 10, 10, 10 ];
value =     [    5,  4,  3,  2 ];
npref =     [              4,  1,             4,          3,      2 ];
good_pref = [  G1, G2, G3, G4, G1, G2, G1, G3, G4, G3, G1, G4, G3, G4 ];
req_pref  = [    6, 6,  6,  6,  6,  6,  4,  6,  6,  6, 6,   5, 10, 10 ];
```

A correct solution is

```
good = [G2, G1, G1, G4, G3];
num = [6, 6, 4, 5, 10];
```

Note that while persons A and C have their second preference and swapping would both give them a first preference, there is not enough of good G1 to allow the swap. In addition while person D has their third preference they wont swap with person E since they have their first preference, nor with persons B or C since good G3 is at best their third preference.

Write a MiniZinc model to solve this problem. You will probably want some form of ranking array. You may want to

1. Write a MiniZinc model to output the ranking array as a new data to be read in.
2. Write constraints in your MiniZinc model to force the ranking array to agree with the given preference data.
3. Build the ranking array as parameter in yout model from the given data.

2 Instructions

Edit the provided `mzn` model files to solve the problems described above. Your implementations can be tested locally by using the *Run* icon in the MiniZinc IDE or by using,

```
minizinc ./modelname.mzn ./datafile.dzn
```

at the command line.