Modeling Discrete Optimization Assignment: **Cryptarithms**

1 Problem Statement

A *cryptarithm* is a mathematical puzzle which requires determining the digit for each letter in an equation. The most famous cryptarithm is SEND + MORE = MONEY. That is we need to determine which digit the letters represent so that,

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

The rules of cryptarithms are:

- Each letter represents a different digit
- The first letter in each word cannot be 0 (otherwise it would not be a proper number)
- The arithmetic equation must hold.

An Example - This Is Easy

Consider the cryptarithm,

$$\begin{array}{ccccc} & T & H & I & S \\ + & & & I & S \\ \hline = & E & A & S & Y \end{array}$$

A MiniZinc model for this problem would be

```
var 1..9: T;
var 0..9: H;
var 1..9: I;
var 0..9: S;
var 1..9: E;
var 0..9: A;
var 0..9: Y;

include "alldifferent.mzn";
constraint alldifferent([T,H,I,S,E,A,Y]);

constraint 1000 * T + 100 * H + 10 * I + S
                              + 10 * I + S
         = 1000 * E + 100 * A + 10 * S + Y;

solve satisfy;
```

Part 1 - This Is Easy

Simply submit the provided `thisiseasy.mzn` model. This will test that MiniZinc is installed and working correctly.

Part 2 - Send More Money

Build a MiniZinc model `smm.mzn` which solves the problem,

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline = & M & O & N & E & Y \end{array}$$

You should determine the unique solution.

Part 3 - Mini Zinc Rockz

Build a MiniZinc model `mzn.mzn` which solves the problem,

$$\begin{array}{cccccc} & & M & I & N & I \\ + & & Z & I & N & C \\ \hline = & R & O & C & K & Z \end{array}$$

with the additional constraint that $Z \leq 5$. For fun, you can check how many different solutions there are.

Part 4 - Send Most Money

Build a MiniZinc model `most.mzn` which solves the problem,

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & S & T \\ \hline = & M & O & N & E & Y \end{array}$$

Rather than simply finding an solution your your model should maximize the value of the number represented by MONEY.

Output Format

If you only declare the necessary variables, e.g. S,E,N,D,M,O,R,Y for part 1, then MiniZincs default output will be in the form required, and you dont need an output item. If you declare other variables you will need to write an output item, to output a value for each variable in .dzn format, e.g.

```
T = 1;
H = 9;
I = 8;
S = 7;
E = 2;
A = 0;
Y = 4;
```

2 Instructions

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

```
mzn-gecode ./modelname.mzn
```

at the command line. Once you are confident that you have solved the problem, submit your solution for grading.

Handin From the MiniZinc IDE, the *coursera* icon can be used to submit assignment for grading. From the command line, `submit.py` is used for submission. In both cases, follow the instructions to apply your MiniZinc model(s) on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions.[1] It may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *programming assignments* section of the course website.

3 Technical Requirements

For completing the assignment you will need MiniZinc 2.0.x and the GECODE 4.4.x solver. Both of these are included in the bundled version of the MiniZinc IDE 0.9.9 (`http://www.minizinc.org`). To submit the assignment from the command line, you will need to have Python 2.7.x installed.

[1]Problem submissions can be graded an unlimited number of times. However, there is a limit on grading of **model submissions**.