"What is essential is invisible to the eye."

The Little Prince, Chapter 21.

1 The Fox (♡ 50 points)

In this exercise, you will practice image processing. Similar to the task during the last lab session during which we brought together a panda and a lion, you should put a red fox next to the Little Prince. You can use any image available on the Internet.

You should send links to two image files and a MATLAB script `fox.m` via email.

2 Kenken (♡ 50 points)

Kenken is a puzzle similar to Sudoku. There is a Kenken generator available at the following URL: http://linuxdingsda.de/∼wintix/kenken/index.php
Assume that the puzzle is represented in a text file in the following format:

```
result operator north east south west
```

Each line encodes information coming from a single square. The first number is the *result* value, if it doesn't exits, it is `-1`. The operator are encoded as follows

-1 no operator (w/o result)

0 no operator (w/ result)

1 multiplication

2 division

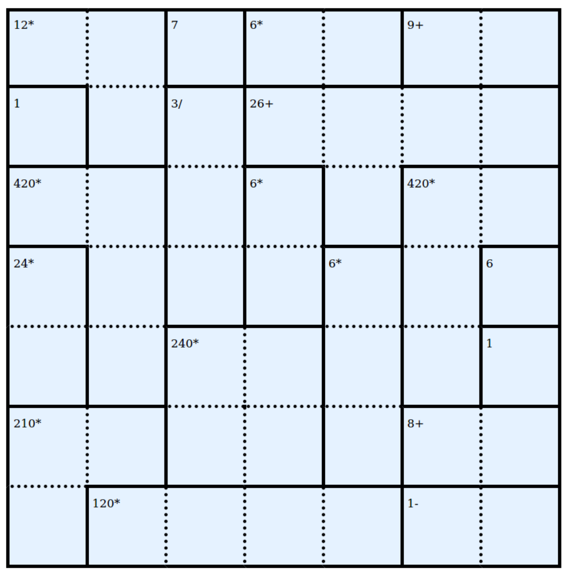
3 addition

4 subtraction

For each direction, if there is a wall, it is represented with `1`. If the square is connected with its neighbor at a particular direction it is represented with `0`. The information for squares are listed from left to right and then from top to bottom.

```
12 1 1 1 1 0
-1 -1 1 0 0 1
7 0 1 1 1 1
6 1 1 1 1 0
-1 -1 1 0 1 1
9 3 1 1 1 0
-1 -1 1 0 1 1
1 0 1 1 1 1
-1 -1 0 1 1 1
3 2 1 1 0 1
26 3 1 1 1 0
-1 -1 1 0 0 0
-1 -1 1 0 1 0
-1 -1 1 0 1 1
420 1 1 1 1 0
-1 -1 1 0 0 1
-1 -1 0 1 0 1
6 1 1 1 0 1
-1 -1 0 1 1 1
420 1 1 1 0 0
-1 -1 1 0 1 1
24 1 1 1 0 1
-1 -1 0 1 0 1
-1 -1 0 1 1 1
-1 -1 0 1 1 1
6 1 1 1 0 1
-1 -1 0 1 0 1
6 0 1 1 1 1
-1 -1 0 1 1 1
-1 -1 0 1 1 1
240 1 1 1 0 0
-1 -1 1 0 0 1
-1 -1 0 1 0 1
-1 -1 0 1 1 1
1 0 1 1 1 1
210 1 1 1 0 0
-1 -1 1 0 1 1
-1 -1 0 1 1 0
-1 -1 0 0 1 1
-1 -1 0 1 1 1
```

```
8 3 1 1 1 0
-1 -1 1 0 1 1
-1 -1 0 1 1 1
120 1 1 1 1 0
-1 -1 1 0 1 0
-1 -1 1 0 1 0
-1 -1 1 0 1 1
1 4 1 1 1 0
-1 -1 1 0 1 1
```



In the class we have seen a Ruby script to scrap the HTML page and create the intermediate format described above. The next task towards writing a solver for Kenken, one has to convert this information into another form which includes the *connectivity* information. Based on the boundary data, it is possible to figure out which squares are connected to each other. For example, we could list the connected squares after each other using their row and column indices. When we need to change from another connected batch, we could list the result and the operator. In order to distinguish the result and the operator from the row and column indices, we could represent the operators with negative numbers only. For example

- -1 no operator (w/o result)
- -2 no operator (w/ result)
- -3 multiplication
- -4 division

-5 addition

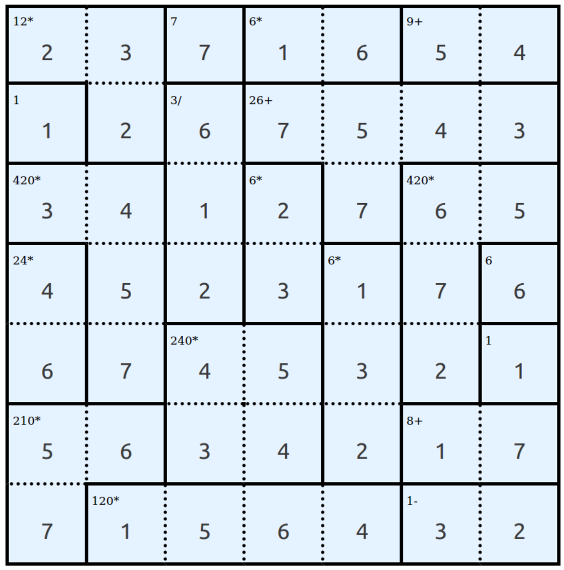
-6 subtraction

The above puzzle could be represented in the following form:

```
1 1
1 2
2 2
12 -3
1 3
7 -2
1 4
1 5
6 -3
1 6
1 7
9 -5
2 1
1 -2
2 3
3 3
4 3
3 -4
2 4
2 5
3 5
2 6
2 7
26 -5
3 1
3 2
4 2
5 2
420 -3
3 4
4 4
6 -3
3 6
4 6
5 6
3 7
420 -3
4 1
5 1
24 -3
4 5
```

```
5 5
6 5
6 -3
4 7
6 -2
5 3
6 3
6 4
5 4
240 -3
5 7
1 -2
6 1
7 1
6 2
210 -3
6 6
6 7
8 -5
7 2
7 3
7 4
7 5
120 -3
7 6
7 7
1 -6
```

Could you write a function which converts between these two formats? You can download more puzzle data using the following link:
http://www.cs1109.info/hws/kenken.zip
The solution of the puzzle above is given as shown in the following figure.



You should send a MATLAB function file named `kenken_converter.m` via email.