UCF Local Contest — September 5, 2009

KenKen Solver

filename: kenken

KenKen is a popular game similar to Sudoku, but more mathematical by nature. The name, loosely translated, means "cleverness squared." Like in Sudoku, you start with an NxN grid, with the rule that each row and column must contain the numbers 1 through N. Furthermore, the grid is split into groups of adjacent cells which contain a number and a mathematical operator (shown by bold lines separating groups). Using the operator, you must place numbers in each group such that when using the operator you get the given number. For example, on a 4x4 grid, given the number 5 and the mathematical operator +, and a group of size 2, you could sum to 5 with the numbers 1 and 4, or with 2 and 3. If the group size was 3, the possible values would be {1,1,3} or {1,2,2}. Note that the order of numbers in the group is not important.

The initial puzzle will not contain any filled cells, and each group will be defined with its number and operator (groups of size 1 contain no operator – these are freebies). The operators used for groups of size greater than 1 are addition (+), subtraction (-), multiplication (*), and division (/). Here is an example of a solved puzzle:

80× 5	4	3 3	5− 1	6	2÷ 2
4	11+ 6	5	1− 2	3	1
9× 3	2 2	3− 1	4	30× 5	6
1	3	11+ 6	5	2÷ 2	4
6 6	8× 1	2	13+ 3	4	8+ 5
10× 2	5	4	6	1 1	3

Note that a group may contain the same number more than once, as long as no row or column contains a repeated number.

Groups using subtraction or division will always be of size 2, and no group will contain more than 10 cells.

The Problem:

Given the description of a KenKen board, solve the puzzle. Each puzzle is guaranteed to be solvable and have a unique solution.

The Input:

You will solve several puzzles. Each puzzle will begin with two positive integers $N \leq 9$ (the size of the board) and $G \leq 52$ (the number of groups in the puzzle). This will be followed by N rows, each containing exactly N letters (starting in column 1), describing the groups on the board.

Each group will be assigned an upper or lower case letter ('a-z' or 'A-Z'), and no two groups will have the same letter identifier (but note that 'a' ≠ 'A'). Each group will be defined by an adjacent set of cells of the same letter, where two cells are adjacent if they share a border to the north, east, south, or west (with no wrap-around top-to-bottom or left-to-right). Following this will be G lines (G is the number of groups in the puzzle). Each line will contain the letter of a group, the number of that group, and the operator of that group, each separated by a space. Each

of the G input lines will describe a unique group. The group identifiers will be the first G lowercase letters; if $G > 26$, the uppercase letters (starting with A and going in order) will also be used. Each mathematical operator will be '+', '-', '*', '/', or '.' (quotes for clarity), where '.' is used to denote a group which has no operator (for groups of size 1). End of input will be denoted by a puzzle with $N = 0$, and should not be processed.

The Output:

For each puzzle, begin with the line "`KenKen Puzzle #`*X*`:`", with X starting at 1. This should be followed by N lines, each containing exactly N digits, giving the solution to this puzzle. Follow each puzzle with a blank line.

(Sample Input/Output on the next page)

Sample Input:

```
2 4
ab
cd
a 1 .
b 2 .
c 2 .
d 1 .
6 18
aabccd
aeeffd
ghiijj
ggkkll
mnnoop
qqnorp
a 80 *
b 3 .
c 5 -
d 2 /
e 11 +
f 1 -
g 9 *
h 2 .
i 3 -
j 30 *
k 11 +
l 2 /
m 6 .
n 8 *
o 13 +
p 8 +
q 10 *
r 1 .
0
```

Sample Output:

```
KenKen Puzzle #1:
12
21

KenKen Puzzle #2:
543162
465231
321456
136524
612345
254613
```

Sudoku Solver

Filename: sudoku

The Problem

Sudoku Puzzles have recently caught on as a hot new item amongst games in newspapers. The game is as follows:

1) You are given a 9x9 grid, with some squares filled in with positive integers in between 1 and 9, inclusive.
2) Your goal is to complete the grid with positive integers in between 1 and 9, inclusive, so that each row, column and mini 3x3 square that is designated contain each integer in the range 1 through 9 exactly once.

Below is an example of a Sudoku puzzle:

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

(taken from www.sudoku.com on 11:45am on 2/1/06)

Here is the puzzle solved:

9	6	**3**	1	**7**	4	**2**	5	**8**
1	**7**	8	3	**2**	5	6	**4**	**9**
2	**5**	**4**	**6**	**8**	**9**	**7**	**3**	1
8	**2**	**1**	4	**3**	7	**5**	**9**	6
4	**9**	6	**8**	**5**	**2**	3	**1**	**7**
7	**3**	**5**	9	**6**	1	**8**	**2**	4
5	**8**	**9**	**7**	**1**	**3**	**4**	**6**	2
3	**1**	7	2	**4**	6	9	**8**	**5**
6	4	**2**	5	**9**	8	**1**	7	**3**

Your program will read in a file of sudoku puzzles, all of which either have no solution or a unique solution. You will output to the screen for each test case, either the filled in game board, or the statement, "No solution possible."

Input Format

The first line of the input file will be a positive integer *n*, representing the number of puzzles to solve in the input file. This will be followed by a blank line. The following 10*n* lines will contain the n cases, with each case taking exactly 10 lines. Within each test case, the i^{th} line will contain the 9 values on the i^{th} row of the unsolved puzzle, in order, from left to right, each separated by spaces. This will be followed by the 10^{th} line that is blank. In particular, all blank entries of the puzzle will be indicated with a 0, and the 9 integer values (all in between 0 and 9, inclusive) on the line each will be separated by a space.

Output Format

The first line for each test case will be of the format:

```
Test case k:
```

where k ranges in between 1 and n, inclusive and represents the test case number. Follow this with a blank line.

If the puzzle has no solution, then the line of output after the blank line will be:

```
No solution possible.
```

If the puzzle has a solution, output it on the following nine lines, with each value separated by a single space. For the puzzle above the output would be:

```
9 6 3 1 7 4 2 5 8
1 7 8 3 2 5 6 4 9
2 5 4 6 8 9 7 3 1
8 2 1 4 3 7 5 9 6
4 9 6 8 5 2 3 1 7
7 3 5 9 6 1 8 2 4
5 8 9 7 1 3 4 6 2
3 1 7 2 4 6 9 8 5
6 4 2 5 9 8 1 7 3
```

Separate the output for each test case with two blank lines.

Sample Input

```
2

0 6 0 1 0 4 0 5 0
0 0 8 3 0 5 6 0 0
2 0 0 0 0 0 0 0 1
8 0 0 4 0 7 0 0 6
0 0 6 0 0 0 3 0 0
7 0 0 9 0 1 0 0 4
5 0 0 0 0 0 0 0 2
0 0 7 2 0 6 9 0 0
0 4 0 5 0 8 0 7 0

0 6 0 1 0 4 0 5 0
0 0 8 3 0 5 6 0 0
2 0 0 0 0 0 0 0 1
8 0 0 4 0 7 0 0 6
0 0 6 0 0 0 3 0 0
7 0 0 9 0 1 0 0 4
5 0 0 0 0 0 0 0 2
0 0 7 2 0 6 9 0 0
0 4 0 5 0 8 0 7 9
```

Sample Output

```
Test case 1:

9 6 3 1 7 4 2 5 8
1 7 8 3 2 5 6 4 9
2 5 4 6 8 9 7 3 1
8 2 1 4 3 7 5 9 6
4 9 6 8 5 2 3 1 7
7 3 5 9 6 1 8 2 4
5 8 9 7 1 3 4 6 2
3 1 7 2 4 6 9 8 5
6 4 2 5 9 8 1 7 3


Test case 2:

No solution possible.
```

Tentaizu!

Filename: `tentaizu`

Tentaizu is a Japanese game whose name means "celestial map." Some people will try to tell you this game is the same as Minesweeper. These people are incorrect.

The game is played on a 7x7 board. Exactly 10 of the 49 squares are each hiding a star. Your task is to determine which squares are hiding the stars. Other squares in the board provide clues: A number in a square indicates how many stars lie next to the square—in other words, how many adjacent squares (including diagonally adjacent squares) contain stars. No square with a number in it contains a star, but a star may appear in a square with no adjacent numbers.

Figure 1 is an example of an initial game board, and Figure 2 is the solution.

1					**3**	
	1			**0**		
			2			
	3					**3**
				1		
			1			**1**

Figure 1

1	*****			*****	**3**	*****
						*****
	1			**0**		
		*****	**2**			*****
	3	*****			*****	**3**
		*****		**1**		*****
			1			**1**

Figure 2

The Problem:

Given the description of a 7x7 Tentaizu board, your task is to find the squares containing the 10 hidden stars. You are guaranteed that each given Tentaizu board will have a unique solution.

The Input:

There will be multiple Tentaizu boards in the input file. The first input line contains a positive integer, *t*, indicating the number of Tentaizu boards to be processed. This will be followed by *t* Tentaizu boards. Each Tentaizu board will contain 7 lines, and each line will contain exactly 7 characters. Each character will be a digit from '0'-'8' or a '.' to indicate an empty square (which may be hiding a star). Each Tentaizu board will be separated by a single blank line.

The Output:

For each Tantaizu board, output the line "`Tentaizu Board #`*x*`:`" where *x* is the board number (starting from 1). Then, print the solved Tentaizu board in the same format as the input, but with a '`*`' at each of the 10 star locations. Leave a blank line after the output of each board.

Sample Input:

```
2
1....3.
.......
.1..0..
...2...
.3....3
....1..
...1..1

...1.3.
12....2
.2.1...
...31..
...2..3
..1....
.......
```

Sample Output:

```
Tentaizu Board #1:
1*..*3*
......*
.1..0..
..*2..*
.3*..*3
..*.1.*
...1..1

Tentaizu Board #2:
...1*3*
12...*2
*2*1...
...31.*
..*2*.3
..1..**
.......
```