SILVER PROBLEMS  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 Three problems numbered 6 through 8  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Problem 6: Hay For Sale [Rob Kolstad (traditional), 2008]  
  
Farmer John suffered a terrible loss when giant Australian cockroaches  
ate the entirety of his hay inventory, leaving him with nothing to  
feed the cows. He hitched up his wagon with capacity C (1 <= C <=  
50,000) cubic units and sauntered over to Farmer Don's to get some  
hay before the cows miss a meal.  
  
Farmer Don had a wide variety of H (1 <= H <= 5,000) hay bales for  
sale, each with its own volume (1 <= V\_i <= C). Bales of hay, you  
know, are somewhat flexible and can be jammed into the oddest of  
spaces in a wagon.  
  
FJ carefully evaluates the volumes so that he can figure out the  
largest amount of hay he can purchase for his cows.  
  
Given the volume constraint and a list of bales to buy, what is the  
greatest volume of hay FJ can purchase? He can't purchase partial  
bales, of course. Each input line (after the first) lists a single  
bale FJ can buy.  
  
PROBLEM NAME: hay4sale  
  
INPUT FORMAT:  
  
\* Line 1: Two space-separated integers: C and H  
  
\* Lines 2..H+1: Each line describes the volume of a single bale: V\_i  
  
SAMPLE INPUT (file hay4sale.in):  
  
7 3  
2  
6  
5  
  
INPUT DETAILS:  
  
The wagon holds 7 volumetric units; three bales are offered for sale with  
volumes of 2, 6, and 5 units, respectively.  
  
OUTPUT FORMAT:  
  
\* Line 1: A single integer which is the greatest volume of hay FJ can  
 purchase given the list of bales for sale and constraints.  
  
SAMPLE OUTPUT (file hay4sale.out):  
  
7  
  
OUTPUT DETAILS:  
  
Buying the two smaller bales fills the wagon.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Problem 7: Patting Heads [Neal Wu, 2008]  
  
It's Bessie's birthday and time for party games! Bessie has instructed  
the N (1 <= N <= 100,000) cows conveniently numbered 1..N to sit  
in a circle (so that cow i [except at the ends] sits next to cows  
i-1 and i+1; cow N sits next to cow 1). Meanwhile, Farmer John fills  
a barrel with one billion slips of paper, each containing some  
integer in the range 1..1,000,000.  
  
Each cow i then draws a number A\_i (1 <= A\_i <= 1,000,000) (which  
is not necessarily unique, of course) from the giant barrel. Taking  
turns, each cow i then takes a walk around the circle and pats the  
heads of all other cows j such that her number A\_i is exactly  
divisible by cow j's number A\_j; she then sits again back in her  
original position.  
  
The cows would like you to help them determine, for each cow, the  
number of other cows she should pat.  
  
PROBLEM NAME: patheads  
  
INPUT FORMAT:  
  
\* Line 1: A single integer: N  
  
\* Lines 2..N+1: Line i+1 contains a single integer: A\_i  
  
SAMPLE INPUT (file patheads.in):  
  
5  
2  
1  
2  
3  
4  
  
INPUT DETAILS:  
  
The 5 cows are given the numbers 2, 1, 2, 3, and 4, respectively.  
  
OUTPUT FORMAT:  
  
\* Lines 1..N: On line i, print a single integer that is the number of  
 other cows patted by cow i.  
  
SAMPLE OUTPUT (file patheads.out):  
  
2  
0  
2  
1  
3  
  
OUTPUT DETAILS:  
  
The first cow pats the second and third cows; the second cows pats no cows;  
etc.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
Problem 8: Jigsaw Puzzles [Rob Kolstad, 2008]  
  
The cows have taken up alphabetical jigsaw puzzles. In these puzzles  
with R (1 <= R <= 10) rows and C (1 <= C <= 10) columns, the edges  
are not funny-shaped cardboard shapes but rather are letters.  
  
Each piece has a serial number and 4 letters (or borders) that must  
be aligned as in a regular jigsaw puzzle. The pseudo-letter '0'  
(the digit 0) will represent a border (and a piece can have several  
borders if it is a corner piece or even the top of column in a,  
e.g., 4x1 puzzle). Below is a set of six pieces (the borders are  
marked with lines instead of '0's) assembled in one way (of many)  
that completes the puzzle:  
  
 +---+ +---+ +---+  
 | 1 c c 3 d d 5 |  
 +-d-+ + a + +-e-+  
  
 +-d-+ +-a-+ +-e-+  
 | 2 b b 4 b b 6 |  
 +---+ +---+ +---+  
  
Note that each edge letter of each piece matches the border letter  
of the piece adjacent to it; the borders appear properly on the top,  
bottom, and sides.  
  
Pieces are represented by a serial number and a clockwise list of  
their four edges (where edges are the letters a..z and 0). Pieces  
might require rotation when placed in the puzzle.  
  
Given a set of pieces, find at least one way to assemble them into  
a puzzle. Test data for puzzles with larger R and C are easier to  
solve because they have a more varied set of edge letters.  
  
PROBLEM NAME: jigsaw  
  
INPUT FORMAT:  
  
\* Line 1: Two space-separated integers: R and C  
  
\* Lines 2..R\*C+1: Each line contains five space-separated entities: an  
 integer serial number and four edge identifiers  
  
SAMPLE INPUT (file jigsaw.in):  
  
2 3  
1 c d 0 0  
2 0 d b 0  
3 c 0 d a  
4 b a b 0  
5 d 0 0 e  
6 0 0 b e  
  
INPUT DETAILS:  
  
Describes the input puzzle although with some of the pieces rotated  
compared to the sample solution.  
  
OUTPUT FORMAT:  
  
\* Lines 1..R\*C: Line R\*(i-1)+j describes the puzzle piece placed a row  
 i, column j with an integer and five space-separated entities:  
 the serial number of the puzzle piece and the four piece edge  
 identifiers in the order top, right, bottom, left.  
  
SAMPLE OUTPUT (file jigsaw.out):  
  
1 0 c d 0  
3 0 d a c  
5 0 0 e d  
2 d b 0 0  
4 a b 0 b  
6 e 0 0 b  
  
OUTPUT DETAILS:  
  
As shown in the diagram in the task text. Other solutions (like  
reflections) are possible; a grading program will check your answer.  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*