Sssssuper big

You are to draw a character-art rendered, squared-off version of the letter S. For any odd integer n bigger than 4, you can draw a reasonable S using the * character that is n characters wide and n characters tall. For ease of counting, you can use the '.' instead of spaces.

Sample input 1: 9
Sample output 1:

```
*********
*
*
*
*********
        *
        *
        *
*********
```

Sample input 2: 5
Sample output 2:

```
*****
*
*****
    *
*****
```

Trending Triples

For any sequence, a subsequence is just a selection from the sequence that keeps the order intact. For example, if the sequence is 3 1 9 6 4 7, then 3 1 7 and 3 1 6 4 are subsequences but 3 4 6 is not a subsequence (because 4 is after 6 in the original) and 3 4 4 7 is not (because the 4 is repeated).

In this problem you will be given a sequence of 4 to 10 positive integers. The end of the sequence will be marked with a -1 (which you should discard). Your job is to print out all subsequences of exactly length three that are strictly increasing.

Sample input 1: 1 2 3 4 5 -1

```
1 2 3
1 2 4
1 2 5
1 3 4
1 3 5
1 4 5
2 3 4
2 3 5
2 4 5
3 4 5
```

Sample input 1: 3 1 9 6 4 7 -1

```
3 6 7
3 4 7
1 6 7
1 4 7
```

A Canny Compression Algorithm

"Edge detection" is an important algorithm in image processing: taking an input image and identifying where there is an area whose brightness changes rapidly. One such algorithm uses the max absolute value of a pixel and its neighbors. Consider the input image below:

15	15	15	15	100	100	100
100	100	100	100	100	100	100
100	100	100	100	100	25	25
175	175	25	25	25	25	25
175	175	25	25	25	25	25

Input image

85	85	85	85	85	0	0
85	85	85	85	85	75	75
75	75	75	75	75	75	75
75	150	150	75	75	75	0
0	150	150	0	0	0	0

Output image

Each output pixel is the max absolute value between the input pixel and any of its (8) neighbors. So, the upper left output pixel is max(|15-15|, |15-100|, |15-100|), which is 85. The pixel in the 4th row, 2nd column is max(|175-100|, |175-100|, |175-100|, |175-175|, |175-25|, |175-175|, |175-175|,|175-25|) which is 150.

Since the images are highly repetitive, they are stored using *run length encoding* (RLE), so instead of saying our first four pixels are 15 and the next 15 are 100, we write 15 4 100 15. Your job in this program is to read in an RLE image, decompress it, compute its output image, and write the output image out in RLE format. The first line of your input will contain the width and height of the graph. Each subsequent line will have a pair of numbers, the first being the value, the second being number of repetitions. The images are guaranteed to be complete (i.e. not too many or too few pixels). Your program must print the output image in RLE format.

Sample input 1 (matches picture above):

```
7 5
15 4
100 15
25 2
175 2
25 5
175 2
25 5
```

Sample output 1:

```
85 5
0 2
85 5
75 10
150 2
75 3
0 2
150 2
0 4
```

Freakin Fractions!

When I was first learning about fractions, one of the most frustrating things for me was that it was difficult to determine which fractions were bigger and which were smaller in any given problem. I was also frustrated that it was so difficult to come up with all the different kinds of fractions of any particular kind, especially since we always had to reduce the fractions to their lowest common denominator.

In this problem you will be given a pair of fractions as individual positive integers. The first fraction will be smaller than the second. Your job is to print out all of the fractions (in reduced form) that have a single digit in the numerator, a single digit in the denominator, and whose value lies between the first and second fraction. Fractions can be output in any order. If there are no fractions in the given range, output "no solution"

Sample input 1: 3 8 4 7
Sample output 1: 2/5 3/7 4/9 1/2 5/9

Sample input 2: 4 9 5 9
Sample output 2: 1/2

Sample input 3: 4 9 1 2
Sample output 3: no solution

Clique busters

Sometimes in a large social group of apparently connected people, there are certain individuals or relationships that form bridges between otherwise unconnected islands. For example, if Bobby, Billy and Bart are friends with each other, and Charlie, Curt and Cecil are friends with each other, if Billy and Curt form a friendship, they've created a link that spans their two islands. In the world of Human Resources, identifying these relationships is very important.

In this problem you'll be given a list of names and a list of relationships (where each relationship is just a pair of names). Relationships are symmetric, which means that if A is a friend of B, then B is a friend of A too. Those relationships are guaranteed to span the set of names: if you take any pair of names, they are either friends, friends of friends, friends of friends of friends, or similar. Your job is to identify if any of the relationships are bridges, i.e. if that relationship ceases to exist, will it break the friendship graph into two distinct pieces.

The first line of the input will contain the number of names n (up to 8 - first names only, no duplicates) and the number of relationships r (up to 36). Next will be the list of n names, one per line. Next come the list of relationships, one per line, where each relationship is specified by a name, followed by a space, and a second name. If there is a critical relationship that is holding two islands together, your program must print the **first such** relationship. If there is no such relationship, your program should print **no critical relationship.**

Sample input 1:
```
6 7
Bobby
Billy
Bart
Charlie
Curt
Cecil
Bobby Billy
Billy Bart
Bart Bobby
Charlie Curt
Curt Cecil
Cecil Charlie
Billy Curt
```

Sample output 1:
```
Billy Curt is a critical relationship
```

Sample input 2:
```
4 4
Andy
Bob
Charlie
Doug
Bob Charlie
Charlie Doug
Bob Andy
Andy Doug
```

Sample output 2:
```
Bob Charlie is a critical relationship
```

Sample input 3:
```
3 3
Shadrach
Meshach
Abednego
Shadrach Meshach
Shadrach Abednego
Meshach Abednego
```

Sample output 3:
```
no critical relationship
```

Not So Random Lottery

The Billings Shell Sorting Club likes runs a small biweekly lottery in which one person wins $100. However, unlike a traditional lottery in which the winner is chosen randomly the winner of the Shell lottery is chosen deterministically. To pick the winner all participants must stand in a circle. Then the president of the club chooses a number, n, and begins walking around the circle starting with person zero. Every n people he removes one person from the circle and begins counting again. The winner is the last person left in the circle.

For example: if the circle has Abby, Bonnie, Joseph, and Clyde and the president choose the number 3 the winner would be determined as follows: First Joseph would be removed (because he is the third person from the start), then Bonnie. After Bonnie comes Clyde and finally, with one person left, Abby would win the lottery.

Input will be on multiple lines as follows. On the first line will be the number of people in the lottery and the number that will be used to determine the winner. Subsequent lines will include the names of people in the lottery.

Example Input 1:
4 3
Abby
Bonnie
Joseph
Clyde

Example Output 1:
Abby wins!

Example Input 2:
6 2
Inky
Pinky
Blinky
Clyde
Mrs. PacMan
Mr. PacMan

Example Output 2:
Mrs. PacMan wins!

Example Input 3:
3 7
Chekhov
Miller
Beckett

Example Output 3:
Beckett wins!

Super knight

In chess, a knight is a piece that always moves in an L-shape pattern, where each move either changes the piece's row by two and column by one, or it changes the piece's column by two and row by one. So a knight at (3,8) can up at (2,6), (1,7), (4,6), or (5,7). The knight's tour says that a knight can start anywhere on a chessboard, and through these 2,1 moves can reach any other location on the board. Consider a variation on this problem where instead of moving 2 in one direction and 1 in another, the knight moves *p* in one direction and *q* in the other, where *p* and *q* are positive integers. If the knight starts at the bottom-left corner of the board - coordinates (1,1) - can the knight (through one or more moves) eventually get to every other square on an 8x8 chessboard. Note: this path need not be unique - if the knight backtracks along the way, all is well.

input: two positive integers p,q
output: a list of all the squares on the board that the piece can ***not*** reach. If the piece can reach all the squares, output "all squares reachable"

example input 1: 2 1
example output 1: all squares reachable
example input 2: 1 1
example output 2: 1,2 1,4 1,6 1,8 2,1 2,3 2,5 2,7 3,2 3,4 3,6 3,8 4,1 4,3 4,5 4,7 5,2 5,4 5,6 5,8 6,1 6,3 6,5 6,7 7,2 7,4 7,6 7,8 8,1 8,3 8,5 8,7

Are there enough burgers?

The Tupper family is getting together for a big barbeque in honor of the late Harrold T. Tupper. Harrold T. used to cook all of the burgers and always had enough for everyone. He used Tupper's formula to calculate how many burgers he needed to buy, but unfortunately he didn't tell anyone what the formula was. Since he won't be attending the gathering this year and the cooking has been left up to the rest of the family, no one is quite sure if they have enough food. They need someone to write a computer program to determine if there is enough burgers for everyone.

Input will be the number of family members to plan for followed by the number of burgers the family bought. Then, on the same line, the number of burger's each of them will eat. Not being afraid of sharing some family members will only each part of a burger and give the rest to somebody else.

Output should be "Enough Burgers" if there is enough burgers for everyone to eat their prefered amount or "*x* burgers needed" where x is the number of burger the family still needs to buy to feed everyone. Keep in mind that you can't buy a fraction of a burger even if the family only needs to buy a fraction more.

For Example:

Input 1:
```
4 10 1.5 3.87 9.8 6
```
Output 1:
```
12 burgers needed
```

Input 2:
```
4 22 1 10 10.1 .8
```
Output 2:
```
Enough burgers
```