

- [Challenges](#) / AA
Rank: 392 Score: 485.71

Plot

In a forest, there were 'x' bunnies, 50% male, and 50% female, all adults. Bunnies doubles every 15 days, 10% of the baby rabbits dies at birth. They mature after 30 days, 30% leave the forest, and rest becomes rabbits. In every 30 days , 25% dies off due to flu. If every bunny dies off, the bunny world ends.

Task

Calculate the final number of bunnies alive after 1 year for any number of initial bunnies, x.

Input

Will be an integer number, the number of initial bunnies. Your program should read from the standard input

Output

Your program should write back to the standard output. When all bunnies die off, write 0 to the standard output.

Please note that percentages are truncated

Example

Test Case 1

444 (input)

0 (output)

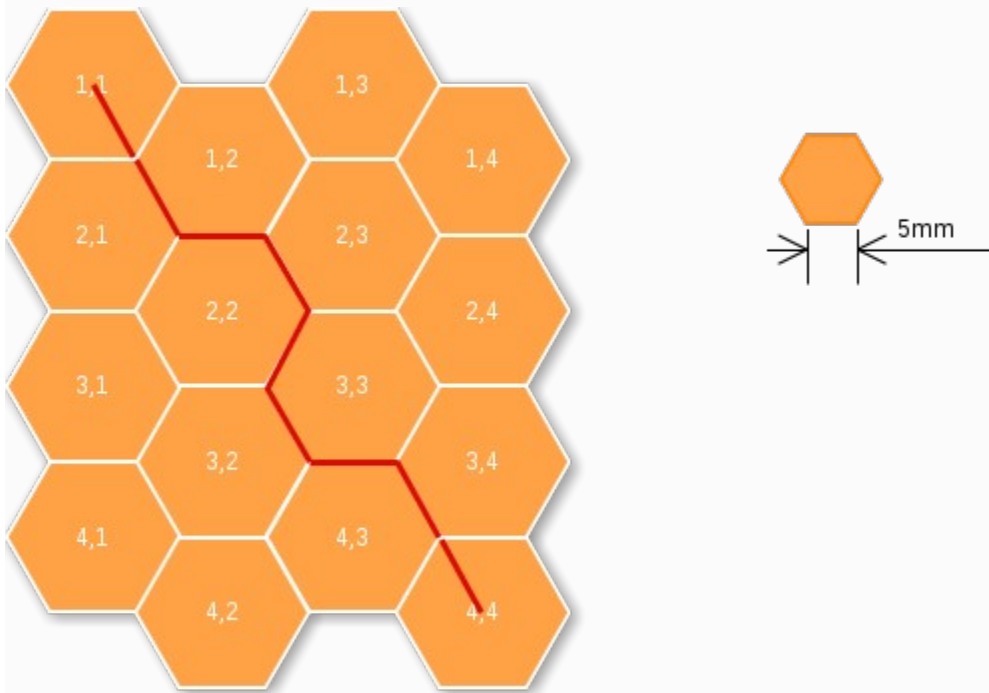
Test Case 2

30000 (input)

56854 (output)

Plot

Ron, the ant, got stuck in a honeycomb. He needs to get out of the honeycomb world.



Task

We know the starting location and the exit point in the honeycomb as x,y co-ordinates. Need to get the minimum distance, the ant has to traverse to get out of the honeycomb, through the edges. maximum size of the bee hive is limited to 12x12. Each edge is 5mm. Input is taken in a format of x1.y1,x2,y2 from standard input and output is expected, the minimum distance to be traveled, as a integer value, in mm. Please check out the above image for more details.

Example

Test Case 1

INPUT: 1.1,4.4

OUTPUT: 45

Test Case 2

INPUT: 1.2,2.4

OUTPUT: 20

Description:

Meet Lakshya, he is one of the best cubers of the world. He has this unique talent to solve a Rubik's cube with his eyes blindfolded! He has recently been experimenting a lot to improve his timing. Lakshya has been following Singmaster's notation of moves to memorize a lot of standard solving techniques. To assist him in his pursuit, you offer to help him. Lakshya has now given you the following description of the tool that he needs:

A tool which will take the initial configuration of rubik's cube (in its solved state) followed by a set of moves, denoted in Singmaster's notation and generate as output the colors of the tiles present on the front face of the rubik's cube.

Constraints:

The input will always be a proper list of moves in Singmaster's notation.

Color code is a string of length 6. Moves can be of any length.

INPUT:

Input consists of two lines, first line will contain the configuration of the cube which has color codes for faces in the following order: Up-Left-Front-Right-Back-Down

Next line will consist of the moves in [Singmaster's notation](#) (Only Basic)

OUTPUT:

A matrix of 3x3 with color code for the tiles in the front face after the moves have been made.

Example

Input

YRBOGW

R2

Output

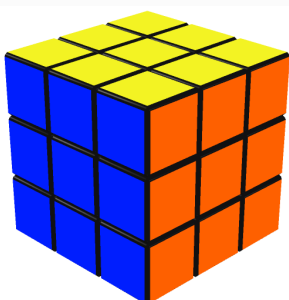
B B G

B B G

B B G

Analysis

Initial state



After R2 (R applied twice)



Input 2

YORBWG

RL'

Output 2

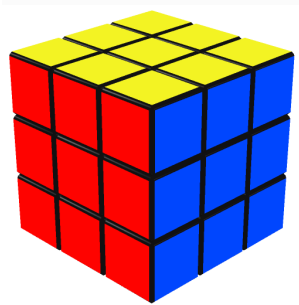
G R G

G R G

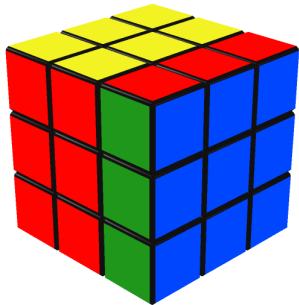
G R G

Analysis 2

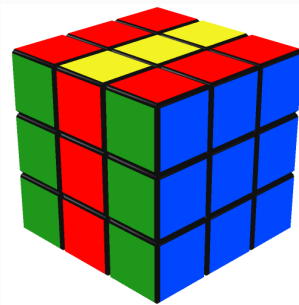
Initial state



After R



After RL' (L applied anti-clockwise)



Orbroid dialer

Problem Statement:

Orbroid is an emerging mobile platform, which is opensource and available for various hardware configurations. Given your inclination towards opensource, you decide to contribute code for this platform and choose the module with following functional specification:

Dialer is the central app of orbroid platform which searches for name/number from the contact book when you start dialing numbers on the T9 keypad. The entries in contact book are internally stored as <firstname>:[lastname]:<number>

where ":" is the field seperator.

The search should consider the T9 text equivalent of entered number. An entry matches if the entered number matches any of firstname, lastname or number field. T9 keypad is shown as follows:

```

-----
- 1 - 2 - 3 -
-... -abc -def -
-----
- 4 - 5 - 6 -
-ghi -jkl -mno -
-----
- 7 - 8 - 9 -
-pqrs-tuv -wxyz-
-----
----- 0 -----
-----|_| -----
-----

```

PS: Any special character apart from space maps to key 1. Space maps to key 0.

Constraints:

firstname can be [a-zA-Z_-. ()]+ and 100 characters in length

lastname can be [a-zA-Z_-. ()]* and 100 characters in length

number can be [0-9]+ and 20 characters in length

Input:

First line contains N, $0 < n < 10^5$, followed by N lines, with each line having format of

First Name:Optional Lastname:Number

Last line containing the string (of numbers) to be searched

Output:

List of matches in contacts, arranged in increasing order of matching position, in format of

First Name:Optional Lastname:Number

In case multiple matches have same matching position then according to the order in which they appear in the input.

In case no matches are found, then the string, "NOT FOUND" has to be printed without quotes.

Example:

Input:

4

Steve:Kilogo:837291091

Mark:Lostworth:428204772

Bill:Laker:256469278

John:Mc millan:7778883929

5646

Output:

John:Mc millan:7778883929

Bill:Laker:256469278

Steve:Kilogo:837291091

Input:

3

Gill:Shaw:61276374888

Mary:Tudor:2818891889

Rajiv:Ghosh:919928388011

8960

Output:

NOT FOUND

Make Change



After a person selects an item from a vending machine, the machine accepts a deposit of coins and/or paper money. If the price of the selected item is less than the deposit, the machine returns change to the user. The machine may use any combination of coins that equals the deposit. For U.S. currency, coins of 0.05, 0.10, 0.25, and 1.00 denominations are returned.

Task

Write a program that:

1. Accepts a price and deposit amount.
2. Computes and writes out the quantity of each type of coin to be returned. The total number of coins is to be minimized; for example, return three 0.25 coins, not seven 0.10 and one 0.05 coins. Assume there are always enough coins to make change for the allowed inputs.

Input

The input is two integers, separated by a space. The first number is the price. The second number is the deposit.

Owing to the absence of built-in money or decimal types in many languages, all amounts (input and output) are represented with positive integers. A valid amount is in the range 0 to 1000 (representing \$0.00 to \$10.00). All amounts must be evenly divisible by 5.

The input is submitted by an end-of-line (e.g., pressing the Enter key.)

Output

Subtract the deposit from the price to determine the return amount. Determine the minimum number of coins needed to equal the return amount. Output the following five numbers, each separated with a space:

If the input does not follow any of the above requirements, output only the string "ERROR".

Example

Input	Output	Note (reference only)
55 100	0 1 2 0	
85 75	ERROR	Price is less than deposit
99 100	ERROR	Price not a multiple of 5
25	ERROR	Missing input

Description

With the new tax system, in Greece, people have to collect receipts and then sum them up. Here, you will help to identify valid receipts from their VAT numbers and then make the sum. A Greek VAT number is 8-digit or 9-digit number. In order to be sure that a VAT number ($A_8A_7A_6A_5A_4A_3A_2A_1$ or $A_9A_8A_7A_6A_5A_4A_3A_2A_1$) is numerically valid we do all the following steps:

- 1) If length of VAT number is 8, then assume that it has a zero digit in front of it, and then continue with a 9 digit string.
- 2) $S = A_1 * 0 + A_2 * 2 + A_3 * 4 + A_4 * 8 + A_5 * 16 + A_6 * 32 + A_7 * 64 + A_8 * 128 + A_9 * 256$
- 3) $Y = S \bmod 11$
- 4) If $Y == 10$ AND $A_1 == 0$, VAT number is numerically valid
- 5) If $Y == A_1$, VAT number is numerically valid
- 6) In any other case, VAT number is not valid

Task

You will be given a list of receipts (VAT number, amount in euro cents) and you are asked to create a program that will identify valid receipts from their VAT number and then return the sum of these receipts.

Input

The input file contains a list of receipts, containing VAT number and amount in euro cents in each line. A single empty line signifies the end of the list.

Output

The output file will contain the sum of all valid receipts, in euro cents, and a new line.

Sample Input

```
094185641 3929
092766360 900
030026340 850
092766360 5500
998198381 590
040933250 800
999517462 250
058302582 1410
052866929 160
998686837 570
998475585 3676
```

Sample Output

```
18635
```


Sample Input with invalid VAT

94185641 3929
92766360 900
30026340 850
92766360 5500
998198381 590
40933250 800
999517463 250
58302582 1410
52866929 160
998686837 570
998475585 3676

Sample Output

18385

Description

We have two tanks that we can fill them with water. We know beforehand the volume of each tank, however, there is no other measurement to see how much water there is in each tank. For example, we can have a tank of 1lt and one of 3lt, and we can't fill the second tank up to 2lt by just looking – we are only sure if we already know how much water already exists in one tank and how much water we are putting in. What we can do is one of the following three(3) valid moves: A) Empty a tank, B) Fill up a tank from a faucet that has unlimited supply, C) Move water to one tank from the other one until the first one fills up or the second one to dry, and if there is remaining water, it is kept or thrown away.

Task

You will be given the volume of each of the two tanks, and also the volume of water we want to find, and you are asked to answer with the minimum number of moves to be done, according to the description, so as to reach the goal, if there is one.

Input

A single line containing 3 numbers, the volume of the first tank, the volume of the second and the goal amount of water we want to find, separated by a single space.

Output

The output will contain just one line containing the minimum number of moves to be done to reach the goal, or the word 'no', in case there is no solution.

Sample Input, 1#

4 5 2

Sample Output, 1#

6

Sample Input, 2#

4 6 3

Sample Output, 2#

no

Robot Tennis

In Robot World a grand championship on tennis is about to take place. All the best robotic tennis players have already started practice for this great event and they are ready to give their best for the Robotic Cup! Unfortunately, due to the bad weather both the tennis field where the tournament was about to take place was ruined and the referee of the game got ill. Can you develop a program that could simulate the tennis field and the coach in order to host the tournament?



Task

Your task is to develop a program that can efficiently simulate a tennis match between two robotic tennis players. The simulation should comply with the following rules of the game:

(Note: please also refer to Fig.1 for a better understanding of the notations and the rules)

1. The tennis court will be a 2 dimensional space comprised of n rows and m columns.
2. At every match, only two robots (**Robot1** and **Robot2** hereafter) can participate.
3. Robot1 can be positioned at **any column** (even the rightmost or the leftmost column) but always at the **first row** of the tennis court whereas Robot2 can be positioned at **any column** but always at the **last row** of the tennis court.
4. On the rightmost and the leftmost side of the tennis court there exist **bouncing** walls which may change the direction of the ball when it collides with them.
5. Moreover, obstacles may exist within the tennis field. Obstacles cannot be positioned at the first and the last row of the tennis field, nor at the leftmost and rightmost columns of the tennis field (i.e. obstacles cannot co-exist neither with robots nor with bouncing walls). Obstacles always alter the direction of the ball.
6. Both robots can **only** move along the horizontal axis (i.e. only to the right or to the left of their current position) and never on the vertical axis.
7. No robot is allowed to stay still. At every step, at first the ball moves and then each robot should moves (on the horizontal axis) towards the ball, meaning that if the ball is positioned at a column i and the robot is at column j , and $i < j$, then the robot should move to the left, whereas it should move to the right if $i > j$. In case that both the ball and a robot happen to be in the same column ($i = j$), the robot should try to move to the **right by default**. If this is not possible (because it is already at the rightmost side of the tennis field and moving to the right means colliding onto the bouncing wall) then it moves to the left side instead.
8. When a robot has the ball, it should always throw it towards its opponent at one of the three possible directions:

- a. *Diagonally to the left (denoted by **L** hereafter),*
- b. *Diagonally to the right (denoted by **R** hereafter) or*
- c. *Straight ahead (denoted by **S** hereafter).*

The direction along which the robot should throw the ball will be decided according to an input sequence that will be provided as input at the beginning of the program. In particular, a sequence of directions (e.g. LRSSR) will be provided as input at the beginning of the program, and every time a Robot has to throw the ball towards its opponent, the next available direction will be chosen (e.g. for the example sequence LRSSR, the first Robot that catches the ball should throw it on the L direction towards its opponent, afterwards the R direction should be selected in order to throw the ball back to the opponent robot, then the S direction and so forth).

1. *The ball should continue to move along the direction defined by the robot until one of the following happens:*
 - a. **The ball collides with a bouncing wall:** *In this case the bouncing wall **may** change the ball direction to its opposite. More specifically, if the ball was moving on the R direction, it changes it to L and vice versa. If the ball was moving on the S direction, the bouncing wall does not change the ball's direction (i.e. it will continue moving straight as if the bouncing wall was not there)*
 - b. **The ball collides with an obstacle:** *In this case, the obstacle **always** changes the movement direction of the ball. In particular, if the ball was moving on the R direction when it collided with the obstacle, it should move on the L direction afterwards and vice versa. In case the ball was moving on the S direction, then the ball should continue moving straight but towards the opposite direction (e.g. if the ball was moving straight heading from Robot1 to Robot2, it should then continue moving straight but heading from Robot2 to Robot1).*
 - c. **The ball reaches a square where a Robot is positioned:** *In this case, the Robot should throw the ball back to the opponent Robot using one of the available directions (L, R or S) and the ball should start moving along this direction afterwards.*
 - d. **The ball reaches the end of the tennis court and no Robot is positioned there:** *In this case, the game ends and the Robot positioned at the other side of the tennis field is nominated winner of the match.*
1. *The game ends when:*
 - a. **The ball reaches the end of the tennis court and no Robot is positioned there:** *In this case, the Robot positioned at the other side of the tennis field is nominated winner of the match.*
 - b. **The ball reaches the end of the tennis court and a Robot is positioned there but there are no remaining directions** *at the provided input sequence so as for this Robot to throw the ball back to its opponent: In this case the game ends without a winner.*

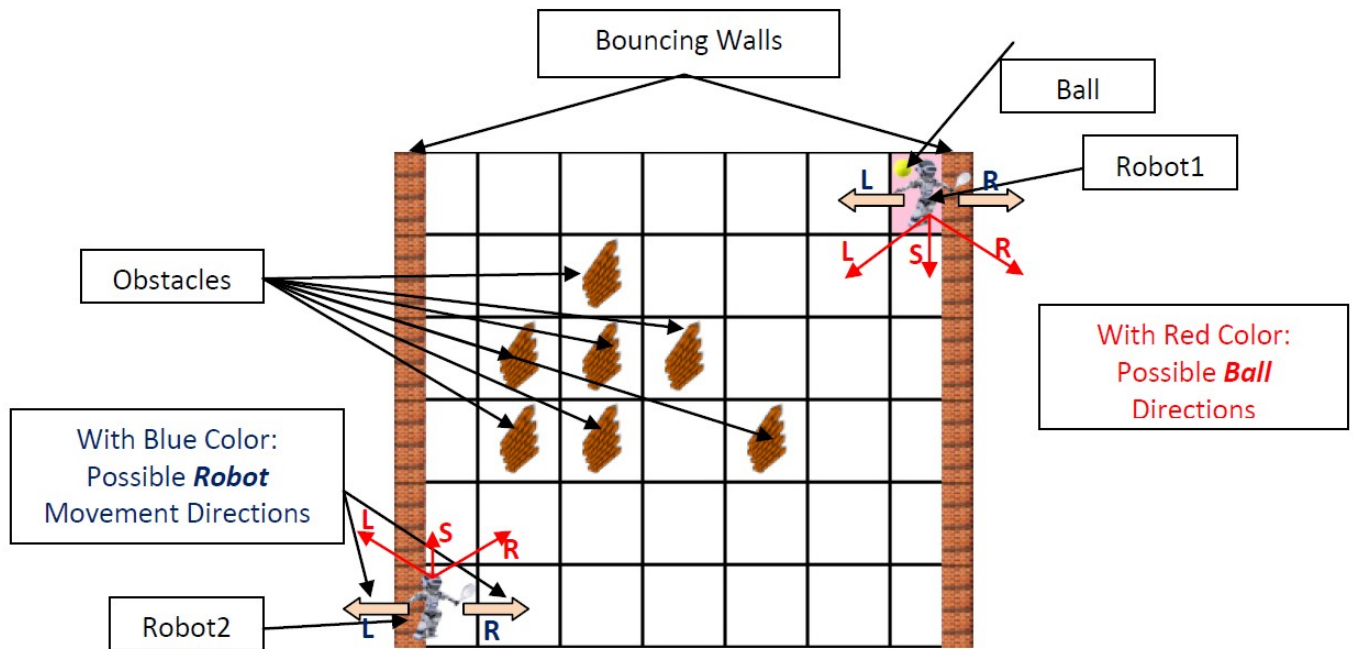
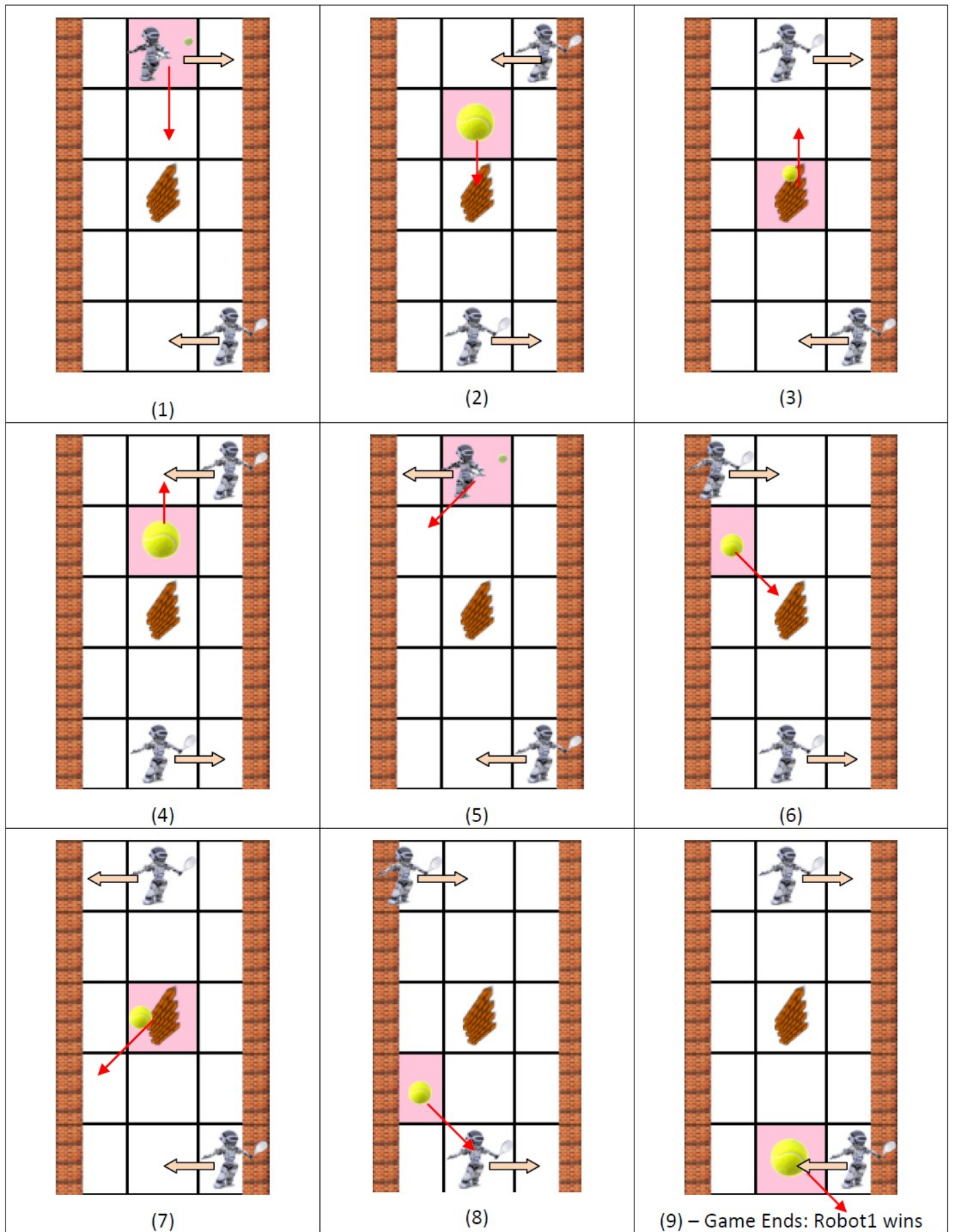


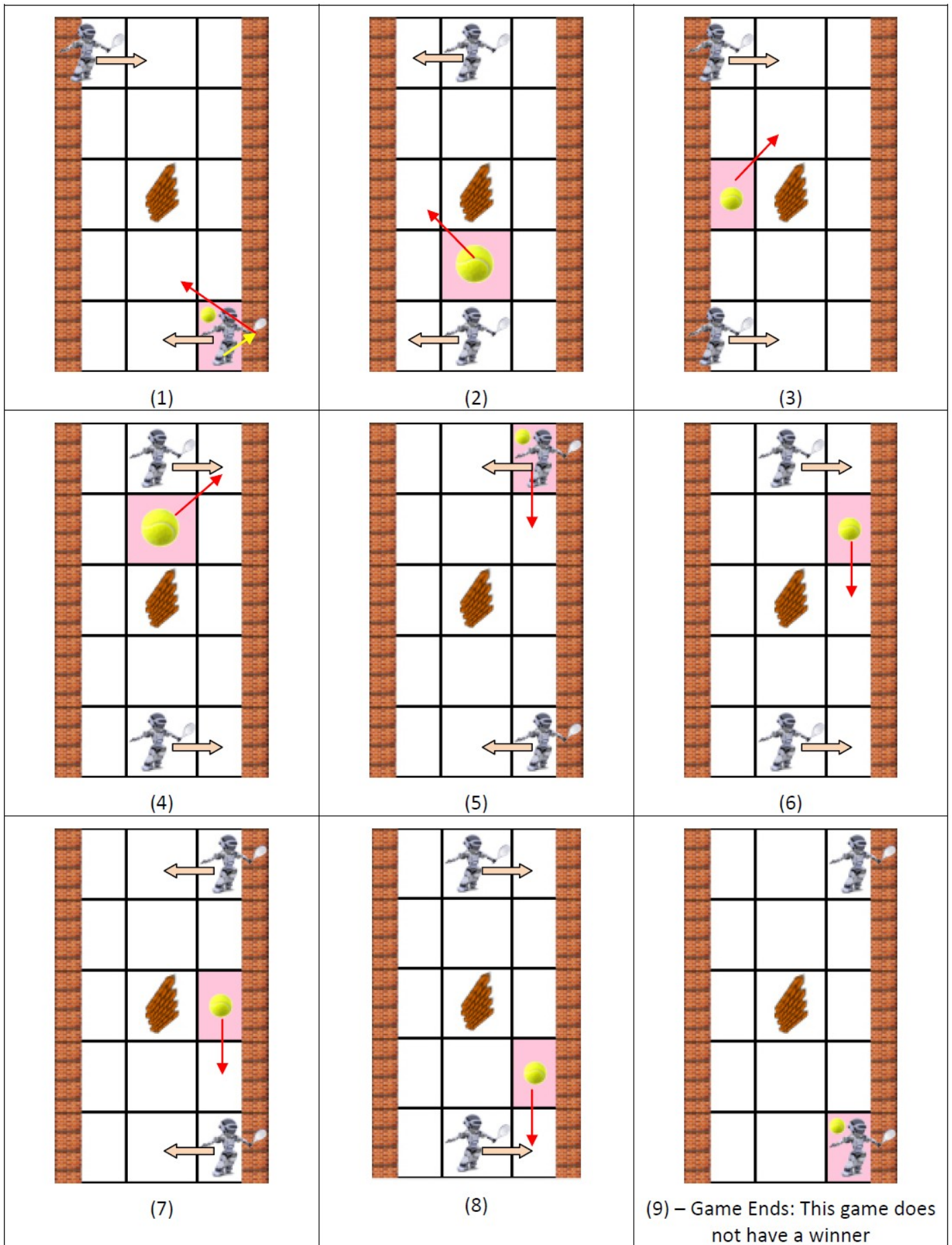
Fig.1: Graphical representation of a Robot Tennis Match.

In order to better explain the rules and the flow of the game, please also consider the following visualizations of a virtual tennis match between the two Robots.

Visualization I: For the Input Sequence {SL} and Robot1 having the Ball at the beginning of the match.
 (Red arrows signify the next position of the ball, whereas pink arrows show where the robot should move at in order to follow the ball's course according to Rule 7)



Visualization II: For the Input Sequence $\{RS\}$ and Robot2 having the Ball at the beginning of the match.



Please note that at Step 1 of Visualization II, although Robot2 sends the ball to the R, since it instantly bounces on the wall its direction changes and thus moves to the L. Moreover, in every step the robots move towards the ball, or they move by default to the right if they are already positioned on the same column with the ball (e.g. Step 1 of Visualization I). In case the robots cannot follow the default right movement rule, then they moved to the left (e.g. Step 7 of

Visualization II). Finally, at Step 9 of Visualization II, the game ends without a winner since there are no available moves for Robot2 to select.

Input

The program receives its input from the standard input stream. The parameters that should be provided as input are the following:

1. Two positive integer numbers n_1 and n_2 (where $1 \leq n_1 \leq 15$ and $1 \leq n_2 \leq 15$) representing the dimensions of the tennis field (i.e. the number of rows and columns respectively). These two numbers will be provided in one line and should be separated by a comma (,) character.
2. A positive integer number r_1_pos ($0 \leq r_1_pos < n_2$) representing the initial position (in terms of column number) of Robot1 (the row number position does not need to be provided since Robot1 is always positioned at the **1st row** of the tennis field).
3. A positive integer number r_2_pos ($0 \leq r_2_pos < n_2$) representing the initial position in terms of column number) of Robot2 (the row number position does not need to be provided since Robot2 is always positioned at the **last row** of the tennis field).
4. A positive integer number $ball_pos$ ($1 \leq ball_pos \leq 2$) representing which robot initially has the ball. More specifically, if $ball_pos$ equals to 1 then Robot1 will initially have the ball, whereas Robot2 will start the game if $ball_pos$ equals to 2.
5. A positive integer number $num_of_obstacles$ ($0 \leq num_of_obstacles \leq \{(n_1 * n_2) - 2(n_1 + n_2) + 4\}$) representing the number of obstacles that will be placed at the tennis field.
6. $num_of_obstacles$ lines should follow representing the position of each obstacle in the tennis field. These positions should be given in the form of pairs of positive integer numbers ob_row, ob_col where $1 \leq ob_row < n_1 - 1$ and $1 \leq ob_col < n_2 - 1$. These pairs should be provided as one pair per line and the two integer values at each line should be separated by a comma (,).
7. A sequence of characters belonging to the set $\{L, S, R\}$ which represents the available moves from which the robots will select at which direction to throw the ball during gameplay. This sequence should be provided in one single line and the characters should not be delimited to each other.

Output

The program should be able to simulate a match between the two competing robots and print to the standard output stream the result of the game, as well as the state of the game (i.e. robot positions, ball position and sequence of movements used) when the game ends. More specifically, the program should output:

1. At the first line:
 - The comment: **"Winner: Robot1"** if Robot1 wins the match
 - The comment: **"Winner: Robot2"** if Robot2 wins the match
 - The comment: **"This game does not have a Winner."** if no robot wins the match
2. At the second line:
 - The comment: **"Robot1 At [x,y]"**, where x and y correspond the row and the column at which Robot1 is positioned when the game ends
3. At the third line:
 - The comment: **"Robot2 At [x,y]"**, where x and y correspond the row and the column at which Robot2 is positioned when the game ends
4. At the fourth line:
 - The comment: **"Ball At [x,y]"**, where x and y correspond the row and the column at which the Ball is positioned when the game ends
5. At the fifth line:

- The comment: **“Sequence: XXXX...”**, where each X stands for a letter belonging to the set {L, R, S}. In this line all moves that were used during the game should be printed at the standard output stream (Beware: only the directions of the ball until the game ended should be printed at this line).

Note: There is a newline character at the end of the last line of the output.

Sample Input 1

```
4,4
0
3
1
1
1,1
LLRSLRSSR
```

Sample Output 1

```
Winner: Robot1
Robot1 At [0,1]
Robot2 At [3,2]
Ball At [3,1]
Sequence: L
```

Sample Input 2

```
8,4
3
2
2
5
1,1
2,1
3,2
5,1
4,2
SRLLRSLLRSSL
```

Sample Output 2

```
Winner: Robot2
Robot1 At [0,2]
Robot2 At [7,1]
Ball At [0,1]
```

Sequence: SR

Digital Calculator

My brother John was so fond of his digital calculator that when the one I had once given him as present broke down, he was very depressed. So, I promised to develop him a program that would actually efficiently simulate such a digital calculator on the computer screen. Can you help me develop such a program?

Task

Your task is to develop a program that can efficiently simulate a digital calculator. The program should be able to receive as input from the standard input stream:

1. Two integer numbers
2. An arithmetic operator (i.e. + for addition, - for subtraction, * for multiplication, / for the integral (Euclidean) division, or % the for the remainder of the integral(Euclidean) division)
3. A character used to draw the numbers on the screen
4. A positive odd integer value representing the size of each digit
5. A positive integer value representing the gap size between the digits

and then draw the arithmetic calculation and the result on the standard output stream. An example output of the program is provided at Fig.1, for the arithmetic operation $(7 * -22)$, using the character '@' to represent digits, selecting a size value equal to 5 and a gap size equal to 3 (dots stand for spaces).

```

.....@@@@@
.....@.
.....@..
.....@...
.....@....
.....
.....@@@@@...@@@@@
***.....@.....@
***.....@@@@@...@@@@@...@@@@@
***.....@.....@....
.....@@@@@...@@@@@
.....
-----
.....
.....@.....@@@@@...@...@
.....@.....@.....@...@
.....@@@@@.....@.....@@@@@...@@@@@
.....@.....@.....@
.....@@@@@...@@@@@.....@

```

Fig. 1. An example of the program output for the arithmetic operation $(7 * -22)$, using the character '@' to represent digits, selecting a size value equal to 5 and a gap size equal to 3

Input

The program should receive as input:

1. One integer value ($-46340 \leq \text{num1} \leq 46340$) representing the first operand
2. Another integer value ($-46340 \leq \text{num2} \leq 46340$) representing the second operand
3. One of the characters +, -, *, /, % representing the operator that will be applied to the operands.
4. One character used for the drawing of the operands.
5. One positive odd integer value ($5 \leq \text{size} \leq 31$) representing the size (i.e. width and height) of each digit
6. Another positive integer value ($1 \leq \text{gaps} \leq 100$) representing the gap size between the digits.

These parameters will be received by the standard input stream, one parameter per line and according to the order described above.

Output

The program should output on the standard output stream the arithmetic calculation between the input operands as well as the calculated result. More specifically:

1. At the first **size** lines, the first integer (num1) number should be drawn
2. Then an empty line (i.e. **full of spaces**) should be drawn
3. At the following **size** lines, the second integer (num2) number should be drawn.
4. Again, an empty line (i.e. **full of spaces**) should be drawn
5. Then, a line **full of dashes (-)** should be drawn, followed by an empty line (i.e. **full of spaces**)
6. Finally, at the last **size** lines, the result of the arithmetic operation should be drawn.

With regards to the horizontal alignment of the output, right side justification should be used. In particular:

1. The first **size** columns correspond to the operator of the calculation
2. Then **gaps** columns will be left blank (i.e. full of spaces)
3. Then, the digits of each operand as well as the ones of the deriving result should be drawn using (**size + gaps**) columns per digit and aligning the digits to the right side (see Fig. 1)

Each digit, should be drawn using the character that was received as input, whereas the operator should be drawn using the symbol that corresponds to the selected mathematical operation (e.g. the '+' should be used to draw the addition operand, whereas the '%' symbol should be used if the remainder operation was selected). The following figure (Fig. 2) graphically depicts all the digits (using the symbol '#') and all possible operators (using the symbol that corresponds to each mathematical operation).

```
.....#.....#####.....#.....#####.....#####.....#####.....#####.....#####
.....##.....#.....#.....#.....#.....#.....#.....#.....#.....#.....#
#####.....#.....#####.....#####.....#####.....#####.....#.....#####.....#####.....#
.....#.....#.....#.....#.....#.....#.....#.....#.....#.....#
.....#####.....#####.....#.....#####.....#####.....#.....#####.....#####.....#####
.....
...../.....%.....%
..+.....***...../.....%.
+++++.....-.....***...../.....%..
..+.....***...../.....%...
...../.....%.....%
```

(a)

(b)

```
.....+.....*****...../.....%.....%%
```

(c)

Fig. 2. Representation of the digits 0-9 (using the symbol '#') and all possible operators (using the symbol that corresponds to each mathematical operation) for size=5 (case a), size=7 (case b) and size=9(case c).

Although the representation of the majority of the aforementioned digits and operators seems to be quite straightforward, special attention has to be paid when drawing:

1. The digit 1, where the diagonal slope should have a height equal to (**size/2**).
2. The digit 7, which is formed out of an horizontal line and a diagonal slope
3. The operator +, which have a height equal to (**size-2**)
4. The operator *, which have a height **and width** equal to (**size-2**)
5. The operator %, where each small circle is represented by a square with edge length equal to (**size/4**)

Note: There is a newline character at the end of the last line of the output.

Sample Input 1

```
-3
-6
*
@
7
2
```

Sample Output 1

```

  @ @ @ @ @ @ @
    @
    @
  @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
    @
    @
  @ @ @ @ @ @ @

  @ @ @ @ @ @ @
**** @
**** @
**** @ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
**** @ @
**** @ @
  @ @ @ @ @ @ @

-----

  @ @ @ @ @ @ @ @
  @ @ @ @
  @ @ @ @
  @ @ @ @ @ @ @
```

@ @ @
@ @ @
@@@@@@@@ @@@@@@@@

Sample Input 2

30120
1215
%
\$
5
3

Sample Output 2

```

    $$$$ $$$$ $  $$$$ $$$$
      $ $ $ $ $ $ $
    $$$$ $ $ $ $$$$ $ $
      $ $ $ $ $ $ $
    $$$$ $$$$ $$$$ $$$$ $$$$

% %      $ $$$$ $ $$$$
%      $$ $ $ $
%      $ $$$$ $ $$$$
%      $ $ $ $
% %      $$$$ $$$$ $$$$ $$$$

-----

    $$$$ $$$$ $$$$
    $ $ $ $ $
    $$$$ $$$$ $ $
      $ $ $ $ $
    $$$$ $$$$ $$$$
```

Cavern Wireless Network

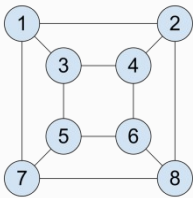
Vangelis the bear finally decided to properly set up the wireless network of his cavern. His cavern is composed of multiple rooms connected by corridors. Due to the thickness of the walls the access to the network is limited only to the neighbouring rooms of the rooms with an access point. Neighbouring rooms, we call rooms that are directly connected by a corridor.

Task

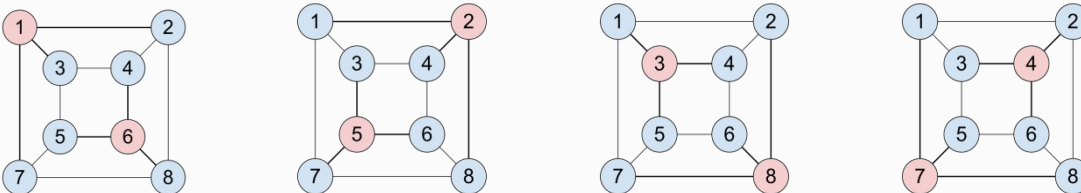
Your task is to study the blueprints of his cavern and show all possible sets of rooms where Vangelis can install an access point, while making sure that:

- each room that has an access point, has no neighbour with an access point,
- all rooms that do not have an access point, have at least one neighbour with an access point,
- he installs as few access points as possible.

Blue Print



All possible access points locations



Input Format

Your program will read N lines (where $0 < N < 100$).

Each line will contain two positive integer numbers (A, B , where $0 < A, B < 100$) separated by one space character. Each line represents the corridor between room A and B .

Output Format

Your program should print all possible sets that match the above criteria.

Each set should be separated by one new line character.

Sets should be printed in ascending order.

Each element of the set should be separated by one space character.

Elements should be printed in ascending order.

Example

Input

1 2

1 3

1 7

2 4

2 8

3 4

3 5

4 6

5 6

5 7

6 8

7 8

Output

1 6

2 5

3 8

4 7

Puzzle

Vangelis the bear and his older brother Mitsos got bored of Sudoku and so decided to create a logic-based, combinatorial, number-placement puzzle of their own to play. The game they created is quite simple:

1. One of the bears, shuffles a series A of N consecutive and unique numbers ($0 < N \leq 20.000$ and $0 < A_i \leq N$)
2. Creates 5 copies of the series and with each copy:
 - a. Takes a random number, that was never moved in a previous copy, and moves it to a random location
3. The 5 variations are then given to the second bear, which is asked to retrieve the original sequence.

Notes: Sometimes a variation could be exactly the same as the original series.

Task Your task is to calculate the original series after receiving the 5 copies.

Input Format

Your program will read $5*N+1$ lines.

The first line will contain number N. Lines 2..N+1: each will contain one number that belongs to the first copy

Lines N+2..2*N+1: each will contain one number that belongs to the second copy. Rest of the lines: similar.

Output Format

Your program should print the original series, one number per line.

Example

Input

```

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

```

Output

5

4

1

3

2

Details

In the above example, the original series is 5,4,1,3,2.

The 5 copies are the following:

- 5,4,1,3,2 -- number 5 was moved to position 1
- 4,5,1,3,2 -- number 4 was moved to position 1
- 1,5,4,3,2 -- number 1 was moved to position 1
- 3,5,4,1,2 -- number 3 was moved to position 1
- 2,5,4,1,3 -- number 2 was moved to position 1

Detecting shapes in a bitmap

Problem statement:

In image analysis, it is common to analyse a bitmap and observe the shapes present in it. For this problem, design an algorithm to detect shapes in a given bitmap. The shapes present in the map shall be from the set Square, Rectangle, Triangle and Parallelogram. In the bitmap each pixel is represented as a bit, 1 - representing black and 0 - representing white. Participants are expected to detect the shapes outlined in black.

Input

The first line will contain the size of the bit map in pixels represented as (Row,Column).

E.g. 6,8 this means a bit map of 6 rows and 8 columns.

The next line will contain a series of decimal digits from 0 to 255 separated by spaces. Each digit will represent a collection of 8 binary bits in the bitmap. IE. 55 represents a binary pattern [00110111](#).

Note: There can be multiple shapes in a bitmap and NO shapes shall intersect. However there can be shapes nested with each other without any intersection.

Output

The shapes present in the bitmap in ascending order of their names, separated by a comma and a space.

Eg. Rectangle, Square, Triangle

Note: There is NO linefeed or space at the end of the output

If any shape repeats, the output should contain as many repetitions as in the bitmap. ie. If there are 2 squares and one triangle, the output shall be Square, Square, Triangle

Example Set 1

Input:

6 8

0 126 66 66 126 0

Output:

Rectangle

Example Set 2

Input:

6 16

0 0 120 120 72 144 73 32 123 192 0 0

Output

Parallelogram, Square

Tangled lunch

Louis Carrol wrote a nice riddle:

"Given that one glass of lemonade, 3 sandwiches, and 7 biscuits, cost 14\$; and that one glass of lemonade, 4 sandwiches, and 10 biscuits, cost 17\$: find the cost of 2 glasses of lemonade, 3 sandwiches, and 5 biscuits?"

We want you to write a program, given the input in sample input 1 to give the right solution, as a rational number in a reduced form and positive denominator: 19/1.

In some cases, the problem cannot be solved; in which case you should output "?".

For example, see sample input 2.

Note that the first input line contains the number of lines in the rest of the input.

Sample 1

input

```
3
1 3 7 14
1 4 10 17
2 3 5
```

output

```
19/1
```

Sample 2

input

```
3
1 1 4 4 10
1 1 5 5 12
1 0 0 0
```

output

```
?
```

Sample 3

input

```
4
1 4 1 4 2 1 3 5 6
2 7 1 8 2 8 1 8 3
3 1 4 1 5 9 2 6 5
0 1 -11 11 -19 54 -43 -11
```

output

```
-77/1
```

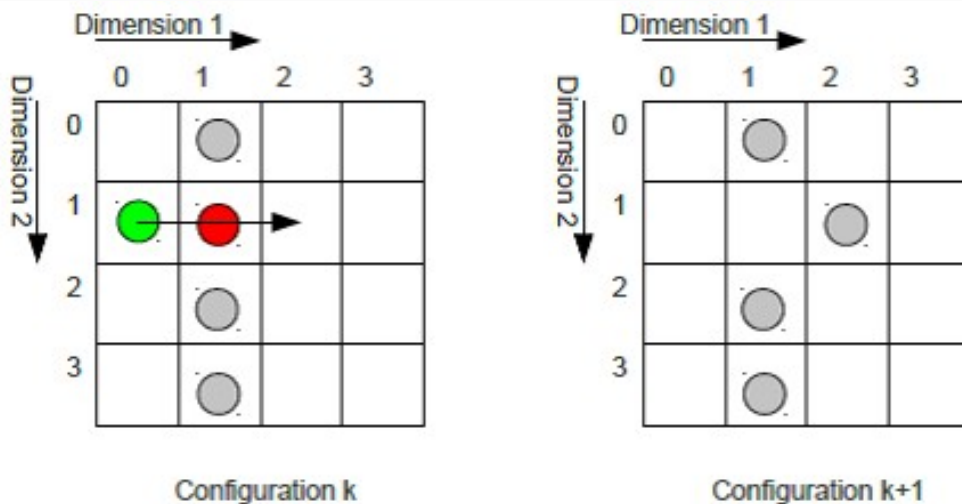
Solution OK.

N-dimensional board game

Consider a *Peg Solitaire* like board game, where a token is moved from a field to an empty field by passing a neighbour token that is in turn removed. A move thus requires an occupied field next to a token and an unoccupied field next to its neighbour in the same direction.

The board has a range of D ($2 < D < 10$) fields in N ($0 < N < 10$) dimensions (thus having D^N fields in total). Each field may be occupied by a token, whose position is denoted with descending dimension: $[p(N) p(N-1) \dots p(2) p(1)]$

The example below illustrates a 2-dimensional board of size $D=4$ and the transition from a configuration k to another configuration $k+1$.



In the example, the token on position $[1\ 0]$ (green) is moved to position $[1\ 2]$ and token $[1\ 1]$ (red) is removed. Obviously, the number of tokens is decremented with every move.

A token can be moved upwards and downwards in any dimension, the maximum number of possible moves of a token is thus $2\ D$.

Every possible move of a configuration branches to a new configuration.

The game is considered to be *lost* if no moves are possible from a configuration and more than one token is left, or to be *won* if a single token is resides in the configuration.

Task

Write a program that reads a start configuration (a single line from *stdin*) and outputs the the number of possible won games (i.e. configurations with only one token left) to *stdout* (0 if no solution is found).

Input / Output format

A start configuration is denoted by a string, starting with size and dimensionality (separated by '-', followed by ':'). Then the occupation of all fields of the configuration is given, while each dimension > 1 is enclosed by braces '{/}' as shown in the table below. An occupied field is indicated by '1', an empty field by '0', separated by space characters. The input is terminated by '\$'.

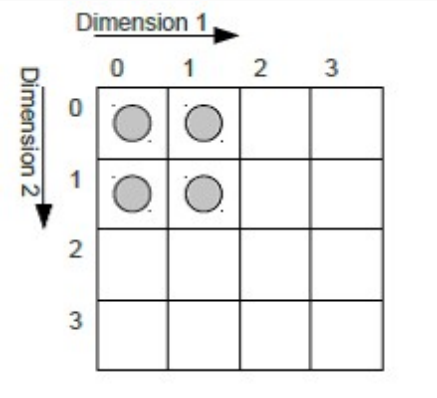
1-D	"D-1:x x x\$"
2-D	"D-2:{x x x}{x x x}{x x x}\$"
3-D	"D-3:{{x x x}{x x x}{x x x}}{{x x x}{x x x}{x x x}}{{x x x}{x x x}{x x x}}\$"

Example input, compare to sketch in the bottom:

4-2:{0 0 1 1}{0 0 1 1}{0 0 0 0}{0 0 0 0}\$

Expected output:

integer number as string.



Example 1

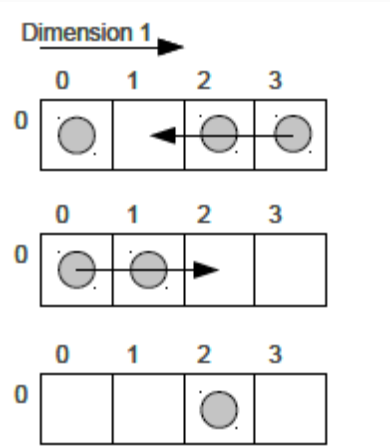
Input:

4-1:1 0 1 1

Output:


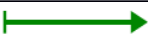




1

One possible solution with two moves as shown on the right.



Wooden railway

A little boy (let's call him Tom) owns a box with a set of wooden rails (the children's toy). There are six rail types as shown in the table below. Each rail has one connector where it starts and at least one connector to further rails (*end points*). The end point locations of a rail are defined by 2D positions (x_e, y_e) and the connector's angle (tangent angle, degrees) with respect to the starting point of the rail. For curved (circular) sections the end point is defined by the rail's radius and the end point's angle.

Sketch	Type	Description	End points (x_e, y_e, angle)
	0	Straight rail, length 20 cm	$(0.2, 0, 0)$
	1	Straight rail, length 30 cm	$(0.3, 0, 0)$
	2	Curved rail, angle 15°	$(x_{e15}, y_{e15}, 15)$ radius = 0.58 m
	3	Curved rail, angle 30°	$(x_{e30}, y_{e30}, 30)$ radius = 0.58 m
	4	Forward switch, angle 15°	$(0.2, 0, 0)$ $(x_{e15}, y_{e15}, 15)$
	5	Reverse switch, angle 15°	$(0.2, 0, 0)$ $(0.2 - x_{e15}, y_{e15}, 165)$

All the rails can be turned on their back, a curved rail can thus be used to turn left or right.

Connecting rails to the end points of other rails (the first rail starts at $(0,0,0)$, the origin) in varying order yields *tracks* of different shape. The end points of rails with no other rail connected to it are considered as *track end points*. Depending on the rails in use and their connections, a track has a number of *track end points* N . This number can be increased only by switches. Because a rail can be connected to several end points at the same time (thus closing a gap between other rails), N can be decrease by any type of rail.

Task

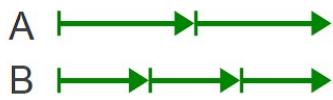
Tom is wondering how many tracks he can create of his set of rails.

Write a program that reads the number of rails of each type N_i from stdin. The program outputs the minimum number of *track end points* and the number of valid different tracks with this property N_i that can be constructed using all the rails in Tom's box.

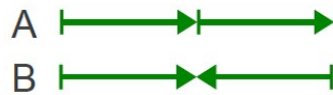
ñ Tracks are considered to be different, if they do not have the same geometrical shape. Two tracks have the same shape, if their connectors have the same positions and orientations.

ñ Tracks with intersecting rails (except the end point connections) are not valid. For simplicity, two rails are assumed to intersect, if the straight lines between their start/end points intersect.

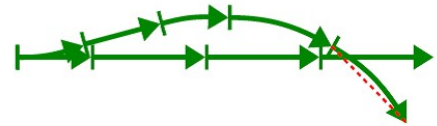
A and B different:



A and B same:



invalid track:



Input / Output format

All numbers are comma separated without any white space characters and on a single line. Rails are given in ascending order $(N_0, N_1, N_2, \dots, N_5)$. The result is expected in the same format: (N_i, N_s) .

Example 1: Two 30° rails

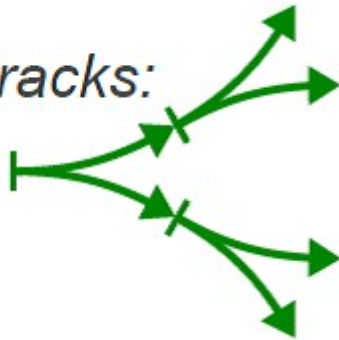
Input:

0,0,0,2,0,0

Output:

1,4

resulting tracks:



Connecting the two curved rails in every possible way yields four tracks with one *track end point* each.

Bob 's Financial Planning

Bob lives in Pecunia. It is a small island city completely governed by the Pecunia City Council (PCC). PCC has decided to encourage foreign investments in the city by waiving several taxes for establishing companies. Consequently, the city has attracted several high skilled workers from across the country. The demand for condos has gone up.

During his career, Bob saved some money. He realized that this is the right time to invest in real estate because the demand is soaring. He decided to purchase condos of different sizes and rent them out. He collected data about condos available for purchase from the classified ads in the local newspaper Pecunia Tribune.

PCC has laid out a plan in order to build infrastructure hand in hand with the amount of investment. According to this, the influx of companies to various parts of the city will be carefully controlled for the next few years. Bob used this data to estimate the prices of the condos in the coming years.

An important factor to keep in mind is property taxes. Bob has to pay a tax to PCC at the end of every month for all the properties he owned during that month. It is a fixed percentage of the estimated value of the property on that date. Also, whenever Bob purchases a property, he needs to pay a fixed percentage of its value as registration fee to the government. Bob is confident that the property tax rate and registration fee percentage will not change during his life time.

Bob has a secure job that guarantees a steady stream of income. He is confident of saving a fixed percentage of his monthly salary (after tax) for investments until his retirement. Also, he gets an annual bonus that can completely be used for investments. He wants to invest his current savings, future savings and the rent he would obtain from his properties. He needs your help in coming up with an investment plan such that his estimated net worth at the time of his retirement is maximized. Net worth is defined as the value of all his properties plus any liquid cash.

Your input is a list of lines. The values in each line are separated by spaces. The significance of each line in the input is explained below.

- Number of years to Bob's retirement [N]
- Bob's current savings [C]
- Percentage of Bob's monthly (after tax) salary that he could use towards investing
- N lines follow with each line containing two numbers – monthly salary for the year (after tax) [Si], annual bonus for the year (after tax) [Bi]. i.e., the first line corresponds to the current year; the next line corresponds to the next year and so on so forth
- Property tax percentage per month [T]
- Property registration fee percentage [F]. This is paid only once when Bob purchases a property
- Number of condos available for purchase [P]
- Each of the P lines that follow contains the details of a condo available for purchase. On each line, there will be one or more 4-tuples (i.e., a sequence of 4 values). Those 4 values represent: year [Yi]; month [Mi]; estimated rent [Ri]; estimated market value [Vi]. For example, the 4-tuple Y1, M1, R1, V1 states that effective from the year Y1 and (the 1st of) month M1, the estimated rent and market value of the condo are R1 and V1 respectively. The estimated rent and market value of the condo are assumed to remain the same until Y2 and M2. Then on, R2 and V2 will apply. Note that the first tuple will always have with Y1 = 1, M1 = 1, and the tuples are chronologically ordered. i.e., the input always has $Y_i \leq Y_{i+1}$ and if $Y_i = Y_{i+1}$ then $M_i < M_{i+1}$.

Output should contain exactly one line: Bob's estimated worth when he retires rounded to 2 decimal points.

Assumptions

- Input is valid.
- All the money is in the same currency. Input numbers are positive floating point values rounded to 2 places after the decimal point. They do not have any formatting (e.g., hundred thousand is input as 100000 or 100000.00 but not as 100,000). Bob's maximum worth at the end of N years is guaranteed to be less than a billion.
- Today is January 1st 2012.
- Bob's annual bonus is deposited on the December 31st of each year. Bob wants to retire at the age of 60. He plans to retire on the December 31st (on or after his 60th birthday). Note that he would get a bonus on that day.
- Bob's salary will remain the same for each month of any given year. It is deposited at the end of each month. His tenants pay each month's rent at the end of it.
- The condos are on a month-to-month lease. Moving out of a condo can happen only on the last day of a month. Moving into a condo can happen only on the 1st of any month i.e., there is no scope for renting a condo for a part of a month.
- Assume that all the properties that Bob did not purchase are available for sale at all the time during the next N years.
- The maximum number of properties available for purchase at any time [P] is less than or equal to 15.
- The banks in Pecunia do not give any interest for keeping Bob's money with them. Also, he is completely averse to borrowing money from people or organizations.
- Bob does property transactions (i.e., purchasing and selling) only at the end of a month.

Example

input

1

651.70

59

12.29 103.89

0.42

2

4

1 1 72.62 741.97 1 6 67.66 646.20 1 7 58.83 563.79 1 10 57.95 526.55 1 12 62.73 656.49

1 1 80.65 832.35 1 11 92.79 951.74 1 12 102.73 975.86

1 1 111.34 976.17 1 3 105.85 895.93 1 7 110.76 920.65 1 10 102.63 887.31 1 11 104.72 1094.79 1 12 94.91 898.43

1 1 67.15 564.28 1 11 65.47 553.74 1 12 52.53 530.26

Output

2248.64

Traffic Light Checker



A three-color traffic light must switch lights in the following order: red, green, yellow, red. To indicate when crossing slowly without a stop is permitted, the yellow signal may be flashed. To indicate when crossing immediately after a full stop is permitted, the red signal may be flashed. Either flash sequence may only be started after the red light is lit and must be followed with a red. All lights are initially off. All sequences must begin a red light. Lighting any light causes all other lights to turn off. No color may be repeated: e.g., red, red is not allowed.

For example, here are some valid sequences:

- Red, Green, Yellow, Red, Green, Yellow, Red
- Red, Green, Yellow, Red, Pause

Here are some invalid sequences:

- Red, Green, Yellow, Red, Yellow
- Green, Caution, Red, Green, Yellow, Red
- Red, Green, Yellow, Yellow, Red, Red

Task

Write a program that checks a sequence of light codes and determines if that sequence follows or violates the traffic light rules.

Input

Read from STDIN a sequence of the follow codes: R (red), Y (yellow), G (Green), P (Pause - flash red), C (Caution - flash yellow), and X (off). Each code must be separated with a space. The entire sequence is submitted by an end-of-line (e.g., pressing the Enter key.) A sequence must have no more than 15 codes in total.

Output

Write to STDOUT an evaluation of the sequence:

<i>ACCEPT</i>	The entire input sequence meets all rules.
<i>REJECT</i>	The input sequence violates any sequence constraint; for example G Y.
<i>ERROR</i>	The input is malformed – undefined code, too short, too long, etc.

Example

Input	Output	Note (reference only)
R G Y R C R G Y R	ACCEPT	
G Y R G Y R	REJECT	Doesn't start with R
R Y G P	REJECT	Invalid sequence
RGY	ERROR	Undefined code
X 8 S	ERROR	Undefined codes

RGYRCRPRGYRGYRGYR

ERROR

Too many codes

A customer may pay one or more open invoices with a single payment. If the payment exactly equals a single invoice or any combination of open invoices, these invoices are paid, reducing their balance to zero. If no match is found, the payment is applied to the oldest invoices.



Task

Write a program that:

1. Accepts a payment amount and open invoice amounts.
2. Pays the invoices, either selecting only those that exactly match the payment amount, or paying as many and as much of the oldest invoices.
3. Outputs resulting balance of the payment and the invoices.

When a payment can pay exactly one or more invoices, the program must apply to only the invoices whose total exactly equals the payment amount. In case of tie, the oldest invoice(s) are paid first. If an exact match cannot be found, the program must apply the payment to the oldest invoice(s) until all of the payment amount is applied. See below for examples.

Input

The input is a series of integers, each separated with a space. The first number is the payment amount, followed by one or more invoice amounts. The program must accept from 1 to 15 invoices.

Owing to the absence of built-in money or decimal types in many languages, all amounts (input and output) are represented with integers.

A valid payment amount is in the range 1 to 1000000.

A valid invoice amount is in the range 1 to 10000.

The entire input sequence is submitted by an end-of-line (e.g., pressing the Enter key.)

Output

The new balance of the payment and invoices are displayed, in the same order as input. Each number is separated with a space.

If the input does not follow any of the above requirements, output only the string "ERROR".

Example

		Note(reference only)
--	--	----------------------

Input	1000 1234 500 500	1000 exactly pays 500 + 500, leaving zero of each. 1234 is unchanged.
Output	0 1234 0 0	
Input	1000 1234 678 282	No exact match, payment applied to oldest.
Output	0 234 678 282	
Input	1000 A B C 5	Invalid input data
Output	ERROR	
Input	1000, 500 500	Invalid input data
Output	ERROR	

Notation:

We use the following notation:

The input to the function is given in variables ("a", "b", etc).

Each Boolean gate gets two inputs and stores its output in the next consecutive letter.

We denote an AND gate by putting its inputs in alphabetical order (the first \leq the second); and an OR gate by reversing the order (the first $>$ the second).

We assume that a capital letter represents the negation of its lower case.

For example, "abBA" computes two gates, the first is AND(a,b) and the second is OR(-a,-b) where -a is the complement of a.

Another example "abAced" is an implementation of the multiplexer function:

$\text{mux}(a,b,c) = b$ if a is true, c otherwise.

Explanation:

d \leq -a AND b; it is an AND gate since a < b < div="">e \leq - (NOT a) AND c; A means (NOT a) and AND since a < c < div="">

f \leq -d or e; it is an OR gate since e > d

And last example, "abABdcCD" computes the XOR and XNOR of two bits.

Task:

You get a function as defined above and you should compute the number of possible inputs which would yield an output of 1 for each computed variable which is not used.

Note: We could have computed the same two outputs for the last example by using "abABdcEE", but then our definition would have given only the second output (XNOR) and not both.

Input:

First line is the number of tests.

Each other line has the number of inputs bits and the function in the notation described above; separated by a single space.

Output:

For each test you should output a single line which contains a list of comma separated numbers. one for each computed, but unused variable in their alphabetical order.

Example 1:

Input:

3

2 abBA

3 abAced

2 abABdcCD

Output:

1,3

4

2,2

Example 2:

Input:

3

4 abcedf

4 dcebfa

3 ABabaAcebc

Output:

1

15

2,0,1,2

</c>><>

- [Challenges](#) /

- AJ

Testing the traversal through different screens

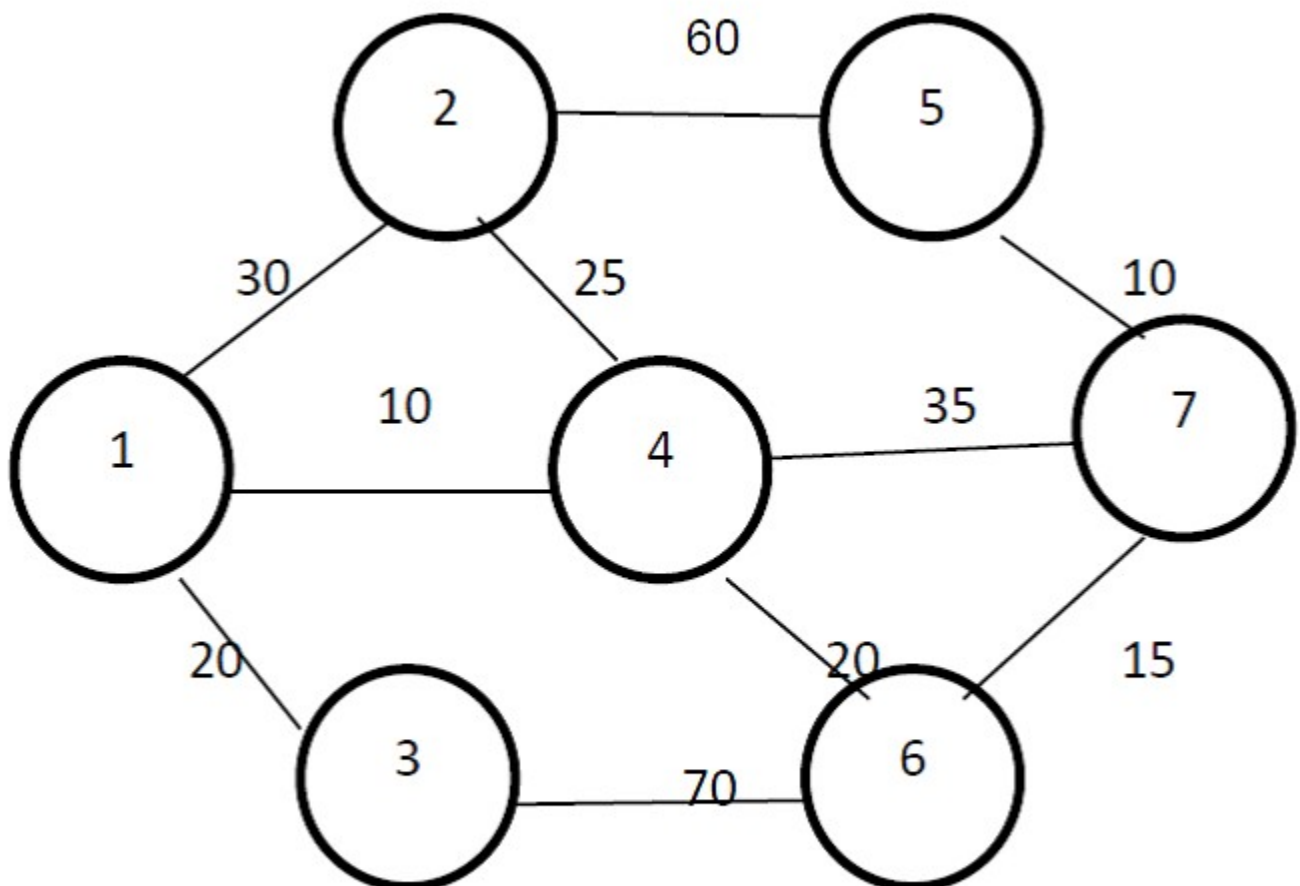
Mr. Ajay is a test expert and he has an innovative approach to testing. His current assignment is to test a particular application which traverses through multiple screens.

One screen can be traversed in multiple ways. The server response time to traverse between screens is different.

The circles in the diagram represent the screens and if the screens are connected by edges, it means that the screen can be traversed from the connecting screen. The numbers associated with the edges represent the minimum response time in microseconds between the screens.

He has to navigate from one screen to a destination screen and return to origin screen, visiting any screen at most once.. What is the fastest way to perform this traversal.

If he has to navigate from 1 to 7, the navigation path he takes is 1-4-6-7-5-2-1



But, Mr. Ajay finds it difficult to find the fastest route himself so he seeks help.

PS: always calculate the path from the first node to the last node and back

Input

The first line of test case will contain two integers: $N(N \leq 100)$ and R representing respectively the number of screens and the connection between screens. Then R lines will follow each containing three integers: C_1 , C_2 and P . C_1 and C_2 are the screen numbers and $P(P > 1)$ is the limit on the minimum server response time to navigate to the screen.

Last line of the input should be the source and the destination screen. Screen numbers are positive integers ranging from 1 to N.

Output

Output the shortest time to traverse from source to destination and back without repeating any screen.

Sample Input

```
7 10
1 2 30
1 3 20
1 4 10
2 4 25
2 5 60
3 6 70
4 7 35
4 6 20
5 7 10
6 7 15
```

Sample Output

```
145
```