Problem A

Finding a root of a polynomial

Input：standard input
Output：standard output
Time limit：3.0 seconds

The Problem

Consider a polynomial f(x). Determine the root of f(x) = 0. You can key in a polynomial f(x) and the root of f(x) = 0 can be output. A root exists during [a, b] while f(a)f(b) ≤ 0. If the equation could not be solved, the message “nil” will be displayed.

The Input

The input consists of several test cases. First line of each case is case title. The second line is the polynomial. The polynomial $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ is typed as a_n*x^n + a_{n-1}*x^n-1 +…+ a_1x + a_0. Blank character has no effect in the expression.

The Output

For each test case, display all the solutions. If the root is not integer, two digits of fractional part are inclusive.
Follow the format of the sample output.

Sample Input

```
2*x^2-4*x+2
x^2+x+1
2*x^2+5*x-3
```

Sample Output

```
1 1
nil
-3 0.50
```

Problem B

Robot Movement

Input：standard input
Output：standard output
Time limit：3.0 seconds

The Problem

A robot needs to move from P1(X1, Y1) to P2(X2, Y2) from an integer xy-coordinate system. The robot can move one step up or right with consuming 2 units of fuel, and move one step down or left with consuming 1 unit of fuel.
There are some obstacles which are not reachable in this plane. An obstacle is expressed as B(X3, Y3). Additionally, there are some wormholes in this system that robot can jump from entry point to exit point without fuel consumption. A wormhole is expressed as W[(X4, Y4),(X5, Y5)], where (X4, Y4) is the entry point and (X5, Y5) is the exit point for this wormhole.
Please design a program to find a path with minimum fuel consumption.

The Input

The positions for P1, P2, obstacles, and wormholes will be given on data1 with specified formats.
If there are additional cases need to be tested, they will be shown on successive data sets with the same definitions as above.

The Output

You need to print out the result of every testing case with the following format:
If the robot cannot move from P1 to P2, please print "The robot cannot make this movement."
If the robot can make the movement and the minimum fuel consumption is XX, please print "The minimum fuel consumption is XX."

Sample Input

P1(4, 1), P2(1, 3), B(3, 0), B(3, 1), B(4, 2), B(5, 1),B(4, 0), W[(2, 1),(3, 4)], W[(6, 6),(8, 2)]
P1(7, 1), P2(5, 5), B(3, 4), B(4, 4), B(5, 4), B(6, 4), B(7, 4), W[(8, 3),(2, 3)], W[(1, 1),(7, 6)]

Sample Output

The robot cannot make this movement.
The minimum fuel consumption is 9.

Problem C

K Smallest Sums

Input：standard input
Output：standard output
Time limit：1.0 seconds

The Problem

You're given k arrays, each array has k integers. There are k^k ways to pick exactly one element in each array and calculate the sum of the integers. Your task is to find the k smallest sums among them.

The Input

There will be several test cases. The first line of each case contains an integer k (2<=k<=750). Each of the following k lines contains k positive integers in each array. Each of these integers does not exceed 1,000,000. The input is terminated by end-of-file (EOF). The size of input file does not exceed 5MB.

The Output

For each test case, print the k smallest sums, in ascending order.

Sample Input

```
3
1 8 5
9 2 5
10 7 6
2
1 1
1 2
```

Sample Output

```
9 10 12
2 2
```

Problem D

Line segments overlapping

Input：standard input
Output：standard output
Time limit： 5.0 seconds

The Problem

Line segments are a very common element in computational geometry. A line segment is the set of points forming the shortest path between two points (including those points). Although they are a very basic concept it can be hard to work with them if they appear in huge numbers unless you have an efficient algorithm.
Given a set of line segments, count how many distinct pairs of line segments are overlapping. Two line segments are said to be overlapping if they intersect in an infinite number of points.

The Input

The first line contains the number of test cases.
Each test case starts with the number n of line segments (1 <= n <= 10000). Then follow n lines consisting of four integers x1, y1, x2, y2 in the range [0, 1000000] each, representing a line segment that connects the points (x1, y1) and (x2, y2). It is guaranteed that a line segment does not degenerate to a single point.

The Output

The output for every test case begins with a line containing "Case No.i:", where i is the number of the test case starting at 1. Then print a single line containing the number of distinct pairs of overlapping line segments followed by an empty line.

Sample Input

```
2
8
1 1 2 2
2 2 3 3
1 3 3 1
10 0 20 0
20 0 30 0
15 0 25 0
50 0 100 0
70 0 80 0
```

1
0 0 1 1

Sample Output

Case No.1:
3
Case No. 2:
0

Problem E

Number Maze

Input：standard input
Output：standard output
Time limit：3.0 seconds

The Problem

Consider a number maze represented as a two dimensional array of numbers comprehended between 0 and 9, as exemplified below. The maze can be traversed following any orthogonal direction (i.e., north, south, east and west). Considering that each cell represents a cost, then finding the minimum cost to travel the maze from one entry point to an exit point may pose you a reasonable challenge.

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

Your task is to find the minimum cost value to go from the top-left corner to the bottom-right corner of a given number maze of size NxM where 1 <= N, M <= 999. Note that the solution for the given example is 24.

The Input

The input file contains several mazes. The first input line contains a positive integer defining the number of mazes that follow. Each maze is defined by: one line with the number of rows, N; one line with the number of columns, M; and N lines, one per each row of the maze, containing the maze numbers separated by spaces.

The Output

For each maze, output one line with the required minimum value.

Sample Input

```
2
4
5
```

---

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

Sample Output

24
15

Problem F

Data Compression Using Run-Length Code

Input：standard input
Output：standard output
Time limit：3.0 seconds

The Problem

Rung-Length Coding is a common approach for data compression, especially where specific patterns repeats frequently. The standard format for Rung-Length Coding is defined as (pattern value, pattern length). For example, MLBMLBMLBMLBBB can be compressed as (MLB, 4) (B, 2) because MLB occurs 4 times and B occurs 2 times. Using 3 bits for the pattern length, write a program to compress a binary bit-string. If the input is not a continuous binary bit-string, display "Invalid input format".

The Input

A continuous binary bit-string with length no longer than 1000

The Output

The compressed binary code with interleaving segments of (0, length of zero) and (1, length of 1)
Note: Because the length is limited to 3 bits, any symbol repeats more than 7 times in a row may require two or more coding segments in a row.

Sample Input

000100000001111111101111111
11100000000000000000011111111111110
MLBMLBMLBMLB

Sample Output

0011 1001 0111 1111 0001 1111
1011 0111 0111 0011 1111 1110 0001
Invalid input format

Problem G

How Many Trees?

Input：standard input
Output：standard output
Time limit：3.0 seconds

The Problem

A binary search tree is a binary tree with root **k** such that any node **v** in the left subtree of k has **label (v)** <**label (k)** and any node **w**in the right subtree of k has **label (w)**> **label (k)**.
When using binary search trees, one can easily look for a node with a given label x: After we compare x to the label of the root, either we found the node we seek or we know which subtree it is in. For most binary search trees the average time to find one of its n nodes in this way is O(log n).
Given a number **n**, can you tell how many different binary search trees may be constructed with a **set** of numbers of size **n** such that each element of the set will be associated to the label of exactly one node in a binary search tree?

The Input

The input will contain a number **i** per line (**1≦i≦~~19~~**1000**)** representing the number of elements of the set.

The Output

You have to print a line in the output for each entry with the answer to the previous question.

Sample Input

```
1
2
3
4
5
```

Sample Output

```
1
2
5
14
42
```

Problem H

Nurikabe

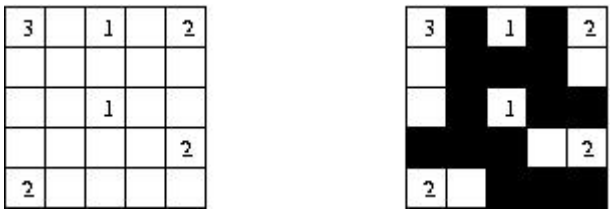
Input：standard input
Output：standard output
Time limit：3.0 seconds

The Problem

The puzzle Nurikabe is played on a grid. Initially, each grid space is either blank or contains a single number. The goal is to shade in some of the blank spaces to satisfy the following constraints:

- Any two shaded spaces are connected by some sequence of adjacent shaded spaces. Two spaces are called adjacent if they share a side.
- For each of the unshaded spaces b, let W_b be the collection of all unshaded spaces that can be reached from b by a sequence of adjacent unshaded spaces. Then, W_b contains exactly one numbered space and that number is exactly the number of spaces in W_b.
- For any unshaded space b, there is a sequence of unshaded spaces starting at b and ending at a space on the edge of the grid where consecutive spaces in this sequence share a side or a corner.
- Finally, in every 2 x 2 subsquare there is at least one unshaded space.

The image shows an example of a Nurikabe puzzle and its solution. Take care to verify all four constraints are satisfied in the solution. To help you understand the third constraint, note that the middle cell containing the number 1 can reach the edge of the grid since it shares a corner with a group of unshaded spaces containing the number 2.



It is known that the problem of determining if a Nurikabe grid can be shaded to satisfy the constraints is NP-complete. Your task is much easier. Given an initial grid and a proposed shading, determine if the shading satisfies the constraints of the Nurikabe puzzle.

Input begins with a single integer t that indicates the number of grids to verify. The first line of each case contains three integers r,c,d where the grid has r rows and c columns ($1 \le r,c \le 100$). Then, d lines follow, each containing three integers r',c',n meaning the grid cell at location (r',c')

contains the positive number *n*. The upper-left grid space has coordinates (0,0), no space receives more than one number, and no two numbered grid spaces will share an edge. Finally, the shading data is specified by *r* strings of *c* characters each (one string per line). The *j*'th character in the *i*'th row of the shading data is '#' if the cell with coordinates (*i*,*j*) is shaded and '.' if that cell is not shaded. Each test case is preceded by a blank line.

For each input case, output a line containing either solved or not solved to indicate whether or not the shading represents a solution to the puzzle.

Sample Input

```
5
5 5 6
0 0 3
0 2 1
0 4 2
2 2 1
3 4 2
4 0 2
.#.#.
.###.
.#.##
###..
..###

5 5 6
0 0 3
0 2 1
0 4 2
2 2 1
3 4 3
4 0 2
.#.#.
.###.
.#.##
####.
..#..
2 3 1
0 0 2
.##
.##
2 2 1
0 0 1
..
##

2 2 2
0 0 1
```

1 1 1
.#
#.

Sample Output

solved
not solved
not solved
not solved
not solved

Problem I

Integer Game

Input：standard input
Output：standard output
Time limit： 3.0 seconds

The Problem

Two players, **S** and **T,** are playing a game where they make alternate moves. **S** plays first.
In this game, they start with an integer **N**. In each move, a player removes one digit from the integer and passes the resulting number to the other player. The game continues in this fashion until a player finds he/she has no digit to remove when that player is declared as the loser.
With this restriction, it's obvious that if the number of digits in **N** is odd then **S** wins otherwise **T** wins. To make the game more interesting, we apply one additional constraint. A player can remove a particular digit if the sum of digits of the resulting number is a multiple of 3 or there are no digits left.
Suppose **N** = 1234. **S** has 4 possible moves. That is, he can remove 1, 2, 3, or 4. Of these, two of them are valid moves.
- Removal of 4 results in 123 and the sum of digits = 1 + 2 + 3 = 6; 6 is a multiple of 3.
- Removal of 1 results in 234 and the sum of digits = 2 + 3 + 4 = 9; 9 is a multiple of 3.

The other two moves are invalid. If both players play perfectly, who wins?

The Input

The first line of input is an integer **T**(**T**<60) that determines the number of test cases. Each case is a line that contains a positive integer **N**. **N** has at most 1000 digits and does not contain any zeros.

The Output

For each case, output the case number starting from 1. If **S** wins then output '**S'** otherwise output '**T'**.

Sample Input

3
4
33
771

Sample Output

Case 1: S
Case 2: T
Case 3: T

Problem J

Making change

Input：standard input
Output：standard output
Time limit： 3.0 seconds

The Problem

Given an amount of money and unlimited (almost) numbers of coins (we will ignore notes for this problem) we know that an amount of money may be made up in a variety of ways. a more interesting problem arises when goods are bought and need to be paid for, with the possibility that change may need to be given. given the finite resources of most wallets nowadays, we are constrained in the number of ways in which we can make up an amount to pay for our purchases--assuming that we can make up the amount in the first place, but that is another story.

The problem we will be concerned with will be to minimise the number of coins that change hands at such a transaction, given that the shopkeeper has an adequate supply of all coins. (the set of new zealand coins comprises 5c, 10c, 20c, 50c, $1 and $2.) thus if we need to pay 55c, and we do not hold a 50c coin, we could pay this as 2*20c + 10c + 5c to make a total of 4 coins. if we tender $1 we will receive 45c in change which also involves 4 coins, but if we tender $1.05 ($1 + 5c), we get 50c change and the total number of coins that changes hands is only 3.

Write a program that will read in the resources available to you and the amount of the purchase and will determine the minimum number of coins that change hands.

The Input

Input will consist of a series of lines, each line defining a different situation. each line will consist of 6 integers representing the numbers of coins available to you in the order given above, followed by a real number representing the value of the transaction, which will always be less than $5.00. the file will be terminated by six zeroes (0 0 0 0 0 0). the total value of the coins will always be sufficient to make up the amount and the amount will always be achievable, that is it will always be a multiple of 5c.

The Output

Output will consist of a series of lines, one for each situation defined in the input. each line will consist of the minimum number of coins that change hands right justified in a field 3 characters wide.

Sample Input

```
2 4 2 0 1 0 0.55
0 0 0 0 0 0
```

Sample Output

~~2~~
3