UVa Online Judge

1079 A Careful Approach

If you think participating in a programming contest is stressful, imagine being an air traffic controller. With human lives at stake, an air traffic controller has to focus on tasks while working under constantly changing conditions as well as dealing with unforeseen events.

Consider the task of scheduling the airplanes that are landing at an airport. Incoming airplanes report their positions, directions, and speeds, and then the controller has to devise a landing schedule that brings all airplanes safely to the ground. Generally, the more time there is between successive landings, the "safer" a landing schedule is. This extra time gives pilots the opportunity to react to changing weather and other surprises.

Luckily, part of this scheduling task can be automated - this is where you come in. You will be given scenarios of airplane landings. Each airplane has a time window during which it can safely land. You must compute an order for landing all airplanes that respects these time windows. Furthermore, the airplane landings should be stretched out as much as possible so that the minimum time gap between successive landings is as large as possible. For example, if three airplanes land at 10:00am, 10:05am, and 10:15am, then the smallest gap is five minutes, which occurs between the first two airplanes. Not all gaps have to be the same, but the smallest gap should be as large as possible.

Input

The input file contains several test cases consisting of descriptions of landing scenarios. Each test case starts with a line containing a single integer n ($2 \leq n \leq 8$), which is the number of airplanes in the scenario. This is followed by n lines, each containing two integers a_i, b_i, which give the beginning and end of the closed interval $[a_i, b_i]$ during which the i-th plane can land safely. The numbers a_i and b_i are specified in minutes and satisfy $0 \leq a_i \leq b_i \leq 1440$.

The input is terminated with a line containing the single integer zero.

Output

For each test case in the input, print its case number (starting with 1) followed by the minimum achievable time gap between successive landings. Print the time split into minutes and seconds, rounded to the closest second. Follow the format of the sample output.

Sample Input

```
3
0 10
5 15
10 15
2
0 10
10 20
0
```

Sample Output

```
Case 1: 7:30
Case 2: 20:00
```

UVa Online Judge

1729 Owllen

Wise owl has got a string S with N ($1 \leq N \leq 10^5$) characters. All the characters of S are lowercase English letters. Now she challenges Fallen to find out a string T of length N such that the length of the **LCS** (Longest Common Subsequence) of S and T is minimum. T also should be consisted of lowercase English letters only.

Now it iss Fallen's problem to find out the string T. But you ou need to print the minimum length of such LCS given that Fallen has found T correctly.

Input

Input file starts with a single integer T ($1 \leq T \leq 50$), T test cases following. Each of the next T test cases has one string S on a line.

Output

For each case print your output in format, '`Case` X`:` Y', on a single line where X denotes the case number starting from 1 and Y denotes the length of the shortest possible LCS.

Sample Input

```
2
ab
efzadeuopqxrvwxaghijklmnbcastbqy
```

Sample Output

```
Case 1: 0
Case 2: 1
```

UVa Online Judge

10271 Chopsticks

In China, people use a pair of chopsticks to get food on the table, but Mr. L is a bit different. He uses a set of three chopsticks – one pair, plus an EXTRA long chopstick to get some big food by piercing it through the food. As you may guess, the length of the two shorter chopsticks should be as close as possible, but the length of the extra one is not important, as long as it's the longest. To make things clearer, for the set of chopsticks with lengths A, B, C $(A \leq B \leq C)$, $(A - B)^2$ is called the "*badness*" of the set.

It's December 2nd, Mr.L's birthday! He invited K people to join his birthday party, and would like to introduce his way of using chopsticks. So, he should prepare $K + 8$ sets of chopsticks(for himself, his wife, his little son, little daughter, his mother, father, mother-in-law, father-in-law, and K other guests). But Mr.L suddenly discovered that his chopsticks are of quite different lengths! He should find a way of composing the $K + 8$ sets, so that the total *badness* of all the sets is minimized.

Input

The first line in the input contains a single integer T, indicating the number of test cases $(1 \leq T \leq 20)$. Each test case begins with two integers K, N $(0 \leq K \leq 1000, 3K + 24 \leq N \leq 5000)$, the number of guests and the number of chopsticks.

There are N positive integers L_i on the next line in non–decreasing order indicating the lengths of the chopsticks $(1 \leq L_i \leq 32000)$.

Output

For each test case in the input, print a line containing the minimal total badness of all the sets.

Note: For the sample input, a possible collection of the 9 sets is:
8,10,16; 19,22,27; 61,63,75; 71,72,88; 81,81,84; 96,98,103; 128,129,148; 134,134,139; 157,157,160

Sample Input

```
1
1 40
1 8 10 16 19 22 27 33 36 40 47 52 56 61 63 71 72 75 81 81 84 88 96 98
103 110 113 118 124 128 129 134 134 139 148 157 157 160 162 164
```

Sample Output

```
23
```

UVa Online Judge

10409 Die Game

Life is not easy. Sometimes it is beyond your control. Now, as contestants of ACM ICPC, you might be just tasting the bitter of life. But don't worry! Do not look only on the dark side of life, but look also on the bright side. Life may be an enjoyable game of chance, like throwing dice. Do or die! Then, at last, you might be able to find the route to victory.

This problem comes from a game using a die. By the way, do you know a die? It has nothing to do with "death." A die is a cubic object with six faces, each of which represents a different number from one to six and is marked with the corresponding number of spots. Since it is usually used in pair, "a die" is a rarely used word. You might have heard a famous phrase "the die is cast," though.

When a game starts, a die stands still on a flat table. During the game, the die is tumbled in all directions by the dealer. You will win the game if you can predict the number seen on the top face at the time when the die stops tumbling.

Now you are requested to write a program that simulates the rolling of a die. For simplicity, we assume that the die neither slips nor jumps but just rolls on the table in four directions, that is, north, east, south, and west. At the beginning of every game, the dealer puts the die at the center of the table and adjusts its direction so that the numbers one, two, and three are seen on the top, north, and west faces, respectively. For the other three faces, we do not explicitly specify anything but tell you the golden rule: the sum of the numbers on any pair of opposite faces is always seven.

Your program should accept a sequence of commands, each of which is either "north", "east", "south", or "west". A "north" command tumbles the die down to north, that is, the top face becomes the new north, the north becomes the new bottom, and so on. More precisely, the die is rotated around its north bottom edge to the north direction and the rotation angle is 90 degrees. Other commands also tumble the die accordingly to their own directions. Your program should calculate the number finally shown on the top after performing the commands in the sequence. Note that the table is sufficiently large and the die never falls off during the game.

Input

The input consists of one or more command sequences, each of which corresponds to a single game. The first line of a command sequence contains a positive integer, representing the number of the following command lines in the sequence. You may assume that this number is less than or equal to 1024. A line containing a zero indicates the end of the input. Each command line includes a command that is one of '`north`', '`east`', '`south`', and '`west`'. You may assume that no white space occurs in any lines.

Output

For each command sequence, output one line containing solely the number on the top face at the time when the game is finished.

Sample Input

```
1
north
3
north
```

```
east
south
0
```

Sample Output

```
5
1
```

UVa Online Judge

11624 Fire!

Joe works in a maze. Unfortunately, portions of the maze have caught on fire, and the owner of the maze neglected to create a fire escape plan. Help Joe escape the maze.

Given Joe's location in the maze and which squares of the maze are on fire, you must determine whether Joe can exit the maze before the fire reaches him, and how fast he can do it.

Joe and the fire each move one square per minute, vertically or horizontally (not diagonally). The fire spreads all four directions from each square that is on fire. Joe may exit the maze from any square that borders the edge of the maze. Neither Joe nor the fire may enter a square that is occupied by a wall.

Input

The first line of input contains a single integer, the number of test cases to follow. The first line of each test case contains the two integers R and C, separated by spaces, with $1 \leq R, C \leq 1000$. The following R lines of the test case each contain one row of the maze. Each of these lines contains exactly C characters, and each of these characters is one of:

- `#`, a wall
- `.`, a passable square
- `J`, Joe's initial position in the maze, which is a passable square
- `F`, a square that is on fire

There will be exactly one `J` in each test case.

Output

For each test case, output a single line containing '`IMPOSSIBLE`' if Joe cannot exit the maze before the fire reaches him, or an integer giving the earliest time Joe can safely exit the maze, in minutes.

Sample Input

```
2
4 4
####
#JF#
#..#
#..#
3 3
###
#J.
#.F
```

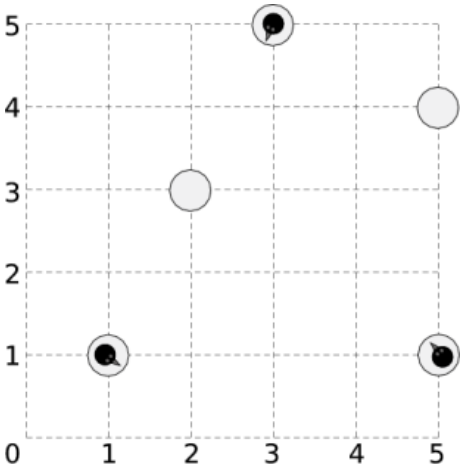
Sample Output

```
3
IMPOSSIBLE
```

UVa Online Judge

12125 March of the Penguins

Somewhere near the south pole, a number of penguins are standing on a number of ice floes. Being social animals, the penguins would like to get together, all on the same floe. The penguins do not want to get wet, so they have use their limited jump distance to get together by jumping from piece to piece. However, temperatures have been high lately, and the floes are showing cracks, and they get damaged further by the force needed to jump to another floe. Fortunately the penguins are real experts on cracking ice floes, and know exactly how many times a penguin can jump off each floe before it disintegrates and disappears. Landing on an ice floe does not damage it. You have to help the penguins find all floes where they can meet.



A sample layout of ice floes with 3 penguins on them.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with the integer N ($1 \leq N \leq 100$) and a floating-point number D ($0 \leq D \leq 100000$), denoting the number of ice pieces and the maximum distance a penguin can jump.
- N lines, each line containing x_i, y_i, n_i and m_i, denoting for each ice piece its X and Y coordinate, the number of penguins on it and the maximum number of times a penguin can jump off this piece before it disappears ($-10000 \leq x_i, y_i \leq 10000$, $0 \leq n_i \leq 10$, $1 \leq m_i \leq 200$).

Output

Per testcase:

- One line containing a space-separated list of 0-based indices of the pieces on which all penguins can meet. If no such piece exists, output a line with the single number '`-1`'.

Sample Input

```
2
5 3.5
1 1 1 1
2 3 0 1
3 5 1 1
5 1 1 1
5 4 0 1
3 1.1
-1 0 5 10
0 0 3 9
2 0 1 1
```

Sample Output

```
1 2 4
-1
```

UVa Online Judge

12333 Revenge of Fibonacci

The well-known Fibonacci sequence is defined as following:

$$\begin{aligned} F(0) &= F(1) = 1 \\ F(n) &= F(n-1) + F(n-2) \quad \forall n \geq 2 \end{aligned}$$

Here we regard n as the index of the Fibonacci number $F(n)$.

This sequence has been studied since the publication of Fibonacci's book *Liber Abaci*. So far, many properties of this sequence have been introduced.

You had been interested in this sequence, while after reading lots of papers about it. You think there's no need to research in it anymore because of the lack of its unrevealed properties. Yesterday, you decided to study some other sequences like Lucas sequence instead.

Fibonacci came into your dream last night. "Stupid human beings. Lots of important properties of Fibonacci sequence have not been studied by anyone, for example, from the Fibonacci number 347746739..."

You woke up and couldn't remember the whole number except the first few digits Fibonacci told you. You decided to write a program to find this number out in order to continue your research on Fibonacci sequence.

Input

There are multiple test cases. The first line of input contains a single integer T denoting the number of test cases ($T \leq 50000$).

For each test case, there is a single line containing one non-empty string made up of at most 40 digits. And there won't be any unnecessary leading zeroes.

Output

For each test case, output the smallest index of the smallest Fibonacci number whose decimal notation begins with the given digits. If no Fibonacci number with index **smaller than 100000** satisfy that condition, output '`-1`' instead — you think what Fibonacci wants to told you beyonds your ability.

Sample Input

```
15
1
12
123
1234
12345
9
98
987
9876
98765
89
32
51075176167176176176
```

```
347746739
5610
```

Sample Output

```
Case #1: 0
Case #2: 25
Case #3: 226
Case #4: 1628
Case #5: 49516
Case #6: 15
Case #7: 15
Case #8: 15
Case #9: 43764
Case #10: 49750
Case #11: 10
Case #12: 51
Case #13: -1
Case #14: 1233
Case #15: 22374
```

UVa Online Judge

12952 Tri-du

Tri-du is a card game inspired in the popular game of Truco. The game uses a normal deck of 52 cards, with 13 cards of each suit, but suits are ignored. What is used is the value of the cards, considered as integers between 1 to 13.

In the game, each player gets three cards. The rules are simple:

- A Three of a Kind (three cards of the same value) wins over a Pair (two cards of the same value).
- A Three of a Kind formed by cards of a larger value wins over a Three of a Kind formed by cards of a smaller value.
- A Pair formed by cards of a larger value wins over a Pair formed by cards of a smaller value.

Note that the game may not have a winner in many situations; in those cases, the cards are returned to the deck, which is re-shuffled and a new game starts.

A player received already two of the three cards, and knows their values. Your task is to write a program to determine the value of the third card that maximizes the probability of that player winning the game.

Input

The input contains several test cases. In each test case, the input consists of a single line, which contains two integers A ($1 \leq A \leq 13$) and B ($1 \leq B \leq 13$) that indicates the value of the two first received cards.

Output

For each test case in the input, your program must produce a single line, containing exactly one integer, representing the value of the card that maximizes the probability of the player winning the game.

Sample Input

```
10 7
2 2
```

Sample Output

```
10
2
```

UVa Online Judge

13076 The traveller squirrel

The popular legend says that in the book *Geography*, written in the first century B.C., the Greek geographer Strabo said that vegetation on the Iberian Peninsula was so dense that a squirrel could cross it from the south to the north jumping from tree to tree without ever touching the ground.

Apparently, Strabo never affirmed such a thing in his book and, in fact, nowadays it is believed that the great achievement of the squirrel wasn't even possible back then.

However, let our imagination fly for a while and think that there was in fact a time in which the number of trees in the Peninsula was large enough to make the achievement possible. Considering that today this is not possible anymore, it is clear that at some point in the past a tree was cut down and caused the separation between the north region and the south region, making it impossible for squirrels to jump from branch to branch.

To simplify the problem slightly, let's assume that the territory is a squared region of size $N \times M$ in which trees (considered of thickness 0) are placed in positions (x, y). The task of the squirrel is to go from the tree in the position $(0, 0)$ to the one in (N, M). You can assume that there is always a tree in both of them. The squirrel can jump from one tree to another if the distance between the two of them does not exceed K units.

The information we have about the territory are the positions of all the trees in the beginning of times. We have to determine the position of the tree that, when cut down, stopped the squirrel from travelling from one part to the other without touching the ground.

Input

Each test case consists of several lines. The first of them contains N, M, K and n ($1 \leq N, M \leq 1,000$; $1 \leq K \leq 10$; $1 \leq n \leq 100,000$), where N, M and K have the meaning described previously and n indicates the number of trees in the territory (without counting the trees in the origin and the destination of the squirrel's journey, which are always present).

After that, there is a line for each of the trees with two integers x, y (the position of the tree). The order in which they are given is the same order followed to cut them down. It is guaranteed that all the positions are inside the territory and that two trees are never placed in the same position.

Output

For each test case, write a single line with the position of the first tree that made the two corners of the field to be unreachable for the squirrel.

If it was never possible for the squirrel to cross the Peninsula from one part to the other, output '`Never had the chance`'.

Sample Input

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

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

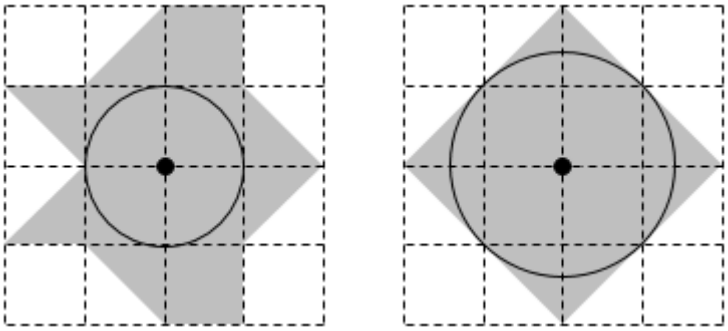
Sample Output

```
2 0
Never had the chance
```

UVa Online Judge

13117 ACIS, A Contagious vIruS

Scientists from REDIS (*REsearch of DISeases*), a famous investigation center in Raccoon City, accidentally caused the mutation of a very contagious virus known as ACIS (*A Contagious vIruS*), just when they were manipulating ACIS' DNA. Raphael, the main researcher at REDIS, was infected with ACIS while he was treating inoculated rats. After that, all persons at REDIS were infected in less than an hour. Immediately he discovered the issue, Raphael contacted the Major, who decided to quarantine the largest possible circular region centered at REDIS that is totally inside Raccoon City, whose boundaries are described with a polygon.



The Major wants to know the maximum radius of such circular region. Can you help him?

Input

The input consists of several test cases. The first line of a test case contains a single integer N indicating the number of vertices of the polygon describing the boundaries of Raccoon City ($3 \leq N \leq 16$). The second line of a test case contains two blank-separated integers x_R and y_R ($0 \leq x_R \leq 50$, $0 \leq y_R \leq 50$) indicating the position (x_R, y_R) where REDIS is located. Then follow N lines: line i contains exactly two blank-separated integers x_i and y_i, where (x_i, y_i) is the position of the i-th vertex of the polygon describing the boundaries of Raccoon City ($0 \leq x_i \leq 50$, $0 \leq y_i \leq 50$). You may assume that there are not two vertices located at the same position, and that REDIS is located inside the polygon excluding its boundaries. The input ends with a line containing a single asterisk ('*').

Output

For each test case, print a single line with a number indicating the radius of the largest possible circular region centered at REDIS that is totally inside Raccoon City. The answer should be formatted and approximated to three decimal places. The floating point delimiter must be '.' (i.e., the dot). The rounding applies towards the *nearest neighbor* unless both neighbors are equidistant, in which case the result is rounded up (e.g., 78.3712 is rounded to 78.371; 78.5766 is rounded to 78.577; 78.3745 is rounded to 78.375, etc.).

Sample Input

```
12
2 2
0 1
1 1
2 0
```

```
3 0
3 1
4 2
3 3
3 4
2 4
1 3
0 3
1 2
4
2 2
0 2
2 0
4 2
2 4
*
```

Sample Output

```
1.000
1.414
```