Problem A : Integer

Given an integer N, express it as the sum of at least two consecutive positive integers. For example:
10=1+2+3+4
24=7+8+9
If there are multiple solutions, output the one with the smallest possible number of summands.

INPUT:
The first line of input contains the number of test cases T. The description of the test cases follow:
Each test case consists of one line containing an integer N (1<=N<=10^9)

OUTPUT:
For each test case, output a single line containing an equation in the format:
N=a+(a+1)+...+b
as in the example. If there is no solution, output a single word `IMPOSSIBLE' instead.

~~~~~~~~
Sample Input:
```
3
8
10
24
```

~~~~~~~~
Sample Output:
```
IMPOSSIBLE
10=1+2+3+4
24=7+8+9
```
~~~~~~~~

Time limit: 3 seconds

Note: The input is always from standard input, the output is always to standard output
~~~~~~~~

Problem B : Objects

Consider a closed world and a set of features that are defined for all the objects in the world. Each feature can be answered with ``yes" or ``no". Using those features, we can identify any object from the rest of the objects in the world. In other words, each object can be represented as a fixed-length sequence of booleans. Any object is different from other objects by at least one feature.

You would like to identify an object from others. For this purpose, you can ask a series of questions to someone who knows what the object is. Every question you can ask is about one of the features. He/she immediately answers each question with ``yes" or ``no" correctly. You can choose the next question after you get the answer to the previous question.

You kindly pay the answerer 10 lei as a tip for each question. Because you don't have surplus money, it is necessary to minimise the number of questions in the worst case. You don't know what is the correct answer, but fortunately know all the objects in the world. Therefore, you can plan an optimal strategy before you start questioning.

The problem you have to solve is: given a set of boolean-encoded objects, minimise the maximum number of questions by which every object in the set is identifiable.

INPUT:
The input is a sequence of multiple datasets. Each dataset begins with a line which consists of two integers, m and n: the number of features, and the number of objects, respectively. You can assume $0 < m \leq 11$ and $0 < n \leq 128$. It is followed by n lines, each of which corresponds to an object. Each line includes a binary string of length m which represent the value (``yes" or ``no") of features. There are no two identical objects.

The end of the input is indicated by a line containing two zeros. There are at most 100 datasets.

OUTPUT:
For each dataset, minimise the maximum number of questions by which every object is identifiable and output the result.

Time limit: 3 seconds

Note: The input is always from standard input, the output is always to standard output

Problem C : Hunter

A good hunter kills two rabbits with one shot. Of course, it can be easily done since for any two points we can always draw a line containing the both. But killing three or more rabbits in one shot is much more difficult task. To be the best hunter in the world one should be able to kill the maximal possible number of rabbits. Assume that rabbit is a point on the plane with integer x and y coordinates. Having a set of rabbits you are to find the largest number of rabbits that can be killed with single shot, i.e. maximum number of points lying exactly on the same line. No two rabbits sit at one point.

INPUT:
The first line in the input contains an integer N (3 <= N <= 200) specifying the number of rabbits.
Each of the next N lines in the input contains the x coordinate and the y coordinate (in this order) separated by a space (?2000 <= x, y <= 2000).

OUTPUT
The output contains the maximal number of rabbits situated in one line that can be killed with one shot.

Time limit: 3 seconds

Note: The input is always from standard input, the output is always to standard output

Problem D : Slides

There are N slides lying on the table. Each of them is transparent and formed as a rectangle. In a traditional problem, one may have to calculate the intersecting area of these N slides. The definition of intersection area is such area which belongs to all of the slides.

But this time I want to take out some one of the N slides, so that the intersecting area of the left N-1 slides should be maximal. Tell me the maximum answer.

INPUT:
The first line of the input contains a single integer T, which is the number of test cases.
The following lines contain the input data for each test case. The first line of each test case contains a single integer N (1<=N<=100) , the number of rectangles. Followed by N lines, each line contains four integers x1 , y1 , x2 , y2 (-10000 <= x1 < x2 <= 10000, -10000 <= y1 < y2 <= 10000) , pair (x1, y1) gives out the bottom-left corner and pair (x2, y2) gives out the top-right corner of the rectangle.

OUTPUT:
There should be one line per test case containing the maximum intersecting area of corresponding N-1 slides.

Time limit: 3 seconds

Note: The input is always from standard input, the output is always to standard output

Problem E: Balanced parantheses

How many balanced sequences of $2n$ parantheses `( )` have at most k consecutive equal characters ?

A sequence of parantheses is balanced if open and closed parantheses are matched one-to-one,
with each closed paranthesis after the matching open one.

Input: a line with the number T of tests, followed by T lines, each with two natural numbers n and k.

Output: T lines, each with the corresponding result.

Example: *Input* / *Output*. Inputs and result fit in 64 bits. Time: 1s

Input	Output
2	2
2 2	1
3 1	

Note: The input is always from standard input, the output is always to standard output

Problem F: Zebra Herd

Time limit: 10s

Description

Zebras are very social animals. Like other members of the horse family, they form groups that tend to stick together and hang out fairly regularly, though not exclusively. (Humans also come to mind in this respect.) Lately, researchers have been trying to understand just how the communities of zebras evolve over time, what triggers changes, and so forth. Of course, all they have to go by is observations of *where* the zebras are over time. From that, we'd like to figure out what are the most natural groups. The assumptions are that (a) if a zebra is part of a group, it tends to spend time close to others in that group, (b) if a zebra is not part of a group, it tends to spend time further away from others in that group, and (c) zebras don't change their group membership very often.

Let's make this more precise. You will be given a sequence of observations of zebras. For each observation time, you will have the exact location of each zebra. The distance between two zebras is exactly their Euclidean (straight-line) distance. We assume that there are exactly two groups of zebras in the herd, and will denote them by two colors. What we want to do is color each zebra either red or blue for each time step, expressing membership to one or the other group. To express assumption (c) above, we will assess a penalty of some given number c every time a zebra changes colors. To express assumptions (a) and (b), we look at the distance $d(i, j)$ between every pair i, j of zebras. If i and j are of the same color, then we assess a penalty of $a \cdot d(i, j)$ for this pair. If i and j are of opposite colors, then we assess a penalty of $-b \cdot d(i, j)$ (i.e., we give a bonus).

Thus, if you are given a proposed labeling of all zebras with either red or blue for each time step, you can compute how good an explanation of zebra activity it is. Your goal is to find the *best* possible labeling, in the sense that it has the smallest possible total penalty. But you'll only need to output the total penalty of the labeling, not the labeling itself.

Input

The first line is the number K of input data sets, followed by the K data sets, each of the following form:

The first line of each data set contains two integers z, t, the number of zebras $2 \leq z \leq 10$, and the number of time steps $2 \leq t \leq 50$. Next comes a line with three floating point numbers $a, b, c \geq 0$, the penalty multipliers. This is followed by t lines, describing zebra positions. Each line contains $2z$ floating point numbers, giving the positions of the zebras in the form $x_1\ y_1\ x_2\ y_2\ \ldots\ x_z\ y_z$. The first line contains the positions at time 1, the second line at time 2, and so forth.

Output

For each data set, output "`Data Set x:`" on a line by itself, where `x` is its number. On the next line, output the minimum penalty that can be achieved by any grouping over time of the zebras, rounded to two decimals. Each data set should be followed by a blank line.

Sample Input/Output

Sample input `zebras.in`

```
1
5 10
1.0 1.0 20.0
0.0 0.0 0.0 0.5 0.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 10.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 0.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 8.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 0.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 0.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 9.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 9.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 9.0 1.0 10.0 0.0 10.0 0.5
0.0 0.0 0.0 0.5 9.0 1.0 10.0 0.0 10.0 0.5
```

Corresponding output

```
Data Set 1:
-476.30
```

Note: The input is always from standard input, the output is always to standard output

Problem G : Decompose

For any three natural numbers, **n, k and x,** we call a ***kx-decomposition*** of number **n** a posibility of writing the **n** number as a sum of **k** non-null numbers so as the difference between any two terms of the sum to be at least **x**.

Given three numbers, **n,k and x**, determine how many distinct ***kx-decompositions*** exist. Two decompositions are considered distinct is they differ by at least one term.

Input:

Read from the standard input three non-null natural numbers, **n, k and x,** separated by one white space.

Output:

Write on the standard output a single value representing the reminder of the division of the number of ***kx-decompositions*** by 10007

Notes:

- 20% of the datasets are going to have $0 < n \leq 200$; 80% are going to have $200 < n \leq 10000$
- $1 \leq x,k \leq n$
- Time limit: 3s

Example

decompose.in	**decompose.out**	**Comments**
20 2 3	8	The number of ***kx-decompositions*** for this case is 8. These are: 1 şi 19; 2 şi 18; 3 şi 17; 4 şi 16; 5 şi 15; 6 şi 14; 7 şi 13; 8 şi 12

Note: The input is always from standard input, the output is always to standard output

Problem H : Fibonacci base (Time limit: 3s)

The well known Fibonacci sequence is obtained by starting with 0 and 1 and then adding the two last numbers to get the next one. For example the third number in the sequence is 1 (1=1+0), the forth is
2 (2=1+1), the fifth is 3 (3=2+1) and so on.

i	0	1	2	3	4	5	6	7	8	9
$Fib(i)$	0	1	1	2	3	5	8	13	21	34

Figure 1 - The first numbers in the Fibonacci sequence

The sequence appears on many things in our life, in nature, and has a great significance. Ancients did not measure the time the way we do today: with past, present and future. The ancients measured time in twinkles, so a man who lived 33 days was considered a very happy man. So it is important for anyone to start counting their real age based on the number of genuine twinkles. But happiness was institutionalized. The Happiness Institution is always located at mystery station 33. All positive integer numbers can be represented as a sum of numbers in the Fibonacci sequence. More than that, all positive integers can be represented as a sum of a set of Fibonacci numbers, that is, numbers from the sequence, without repetition. For example: *13* can be the sum of the sets *{13}*, *{5,8}* or *{2,3,8}* and *17* is represented by *{1,3,13}* or *{1,3,5,8}*. Since all numbers have this property (do you want to try to prove this for yourself?) this set could be a nice way to use as a "base" to represent the number. But, as we have seen, some numbers have more than one set whose sum is the number. How can we solve that? Simple! If we add the constraint that the sets cannot have two consecutive Fibonacci numbers, than we have a unique representation for each number! This restriction is because the sum of any two consecutive Fibonacci numbers is just the following Fibonacci number.

Now that we know all this we can prepare a nice way to represent any positive integer. We will use a binary sequence (just zeros and ones) to do that. For example, *17 = 1 + 3 + 13* (remember that no two consecutive Fibonacci numbers can be used). Let's write a zero for each Fibonacci number that is not used and one for each one that is used, starting at the right. Then, *17 = 100101*. See figure 2 for a detailed explanation. In this representation we should not have zeros at the left, this is, we should only write starting with the first one. In order for you to understand better, note that in this scheme, not using two consecutive Fibonacci numbers means that the binary sequence will not have two consecutive ones. When we use this representation for a number we say that we are using the Fibonaccimal base, and we write it like *17 = 100101 (fib)*.

17 =	1	0	0	1	0	1
13+3+1 =	13	8	5	3	2	1

Figure 2 - Explaining the representation of 17 in Fibonaccimal base

Given a set of numbers in decimal base, your task is to write them in the Fibonaccimal base.

Input

The first line of input contains a single number N, representing the quantity of numbers that follow ($1 \leq N \leq 500$).

Than follow exactly N lines, each one containing a single positive integer smaller than 100 000 000. These numbers can come in any order.

Output

You should output a single line for each of the N integers in the input, with the format '*DEC BASE* = *FIB BASE* (fib)'. *DEC BASE* is the original number in decimal base and *FIB BASE* is its representation in Fibonaccimal base. If the FIB_BASE value for 33 can be identified anywhere in the expression of the FIB_BASE number, "(fib)" must be uppercase "(FIB)". See the sample output for an example.

Sample Input

```
10
1
2
3
4
5
6
7
8
9
10
```

Sample Output

```
1 = 1 (fib)
2 = 10 (fib)
3 = 100 (fib)
4 = 101 (fib)
5 = 1000 (fib)
6 = 1001 (fib)
7 = 1010 (fib)
8 = 10000 (fib)
9 = 10001 (fib)
10 = 10010 (fib)
```

Note: The input is always from standard input, the output is always to standard output