Algospot.com 2010 ACM-ICPC Daejeon Regional Warm-up Contest

October 23rd, 2010

Contestant Guidelines

Important Rules

- Programs must read from **Standard Input**, and write to **Standard Output**. For specific guidelines about I/O, please refer to Language Specific Guidelines.
- It is not required to print all the answers at once even if there are multiple test cases; your program may print the answer for each test case one by one. For more details, please refer to Language Specific Guidelines.
- Please pay attention to your program's **Return Code**. If your program returns a non-zero return code at termination, your submission will result in **"No – Runtime Error"**.
- Your program's file name must be alpha-numeric. In particular, whitespace(s), tabs, special characters, Korean letters, etc. in your file name may cause our judge software to judge your program incorrectly, and you may receive **"No – Compilation Error"** or other error message.
- In order to make this contest as realistic as possible, please do refrain from using **the internet** or any reference materials that are prohibited in ACM ICPC contest rules.
- If you have doubts or questions about the problems, please use the judge system PC^2's **Clarification** feature. We have the right not to answer to your question by replying with a message "**No response, read the problem statement**". Such questions include, but are not limited to, the following:
  - *How many test cases do you have?:* there will be a number of test cases that will guarantee that the *intended* solution will run in a *reasonable* amount of time
  - *What is the time limit?:* see the question and answer above
  - *Why is my submission* **No – Wrong Answer***?:* that is because the answer your program generated is wrong. We are very confident that our answers are correct as they are thoroughly verified
  - *Can I assume that this and that hold in this problem?:* You may assume anything you would like as long as your program produces the correct answer
  - *My program runs well on my machine, but why do I receive* **"No – Compilation Error"***?:* It is likely that the compiler you are using and the compiler our system uses differ in their versions. In particular, if you are using VC++ 6.0, this happens more than often. Please refer to Language Specific Guidelines.

Run Results

Yes Your program passed all test cases we have within the time limit, congratulations!

No – Compilation Error Your program did not compile successfully on our system.

No – Run-time Error Your program did not terminate normally while running.

No – Time–limit Exceeded Your program did not terminate within the time limit; we do not reveal the time limits.

No – Wrong Answer The answer your program generated did not match the correct answer of ours.

No – Excessive Output Your program generated an excessive amount of output and thus was forcefully killed.

No – Output Format Error Your program generated an answer that did not follow the output format specified in the problem.

No – Other – Contact Staff For a different reason than the ones above, your program was not run. If you receive this message, you must contact the staffs via **Clarification**.

If your program is incorrect in multiple ways, you may receive any one of the corresponding error messages. For instance, if your answer is both incorrect and uses a wrong format, you may receive "**No – Output Format Error**" or "**No – Wrong Answer**" (or it could be something else if there are other errors, too).

Language Specific Guidelines

C

We support the following compiler and compile options.

GNU C 4.5.0 (MinGW) `-O2 -lm`

Microsoft C++ Visual C++ 2010 (16.00) `/O2`

The following example code gets the number of test cases first, and for each test case, it outputs the sum of two integers given.

```
#include <stdio.h>

int main()
{
        int t;
        scanf("%d", &t);
        for (int i = 0;i < t;i++)
        {
                int a, b;
                scanf("%d␣%d", &a, &b);
                printf("%d\n", a + b);
        }
        return 0;
}
```

C++

We support the following compiler and compile options.

GNU C 4.5.0 (MinGW) `-O2 -lm`

Microsoft C++ Visual C++ 2010 (16.00) `/O2`

The following example code gets the number of test cases first, and for each test case, it outputs the sum of two integers given.

```
#include <iostream>

using namespace std;

int main()
{
	int t;
	cin >> t;
	for (int i = 0;i < t;i++)
	{
		int a, b;
		cin >> a >> b;
		cout << a + b << endl;
	}
	return 0;
}
```

Java

We support the following compiler and compile options.

Oracle Java JDK 6 Update 22

The following example code gets the number of test cases first, and for each test case, it outputs the sum of two integers given.

Please note that the name of your Java file and the name of the class must be **identical** due to the policy of the Java language.

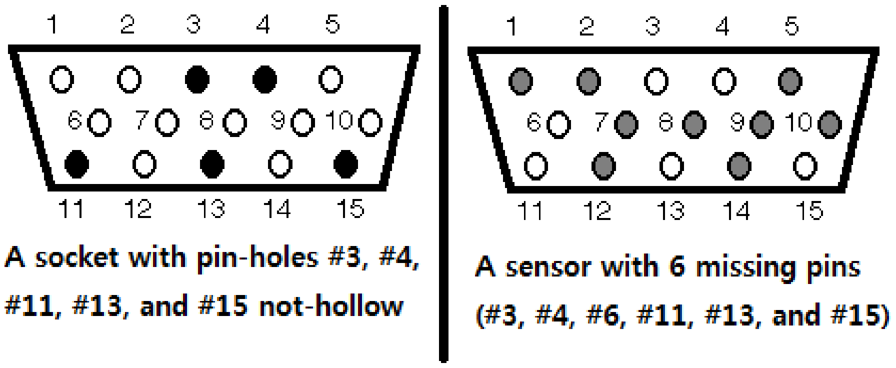
```
import java.util.*;

public class Adder
{
	public static void main(String[] args)
	{
```

```
                Scanner sc = new Scanner(System.in);
                int t = sc.nextInt();
                for (int i = 0;i < t;i++)
                {
                        int a = sc.nextInt();
                        int b = sc.nextInt();
                        System.out.println(a + b);
                }
        }
}
```

Problem A. Plug-in

Consider a socket having X rows of pin−holes, and a sensor having X rows of pins. Some pin−holes on the socket may not be hollow, which do not allow a pin to be plugged in (because a pin can only be plugged into a hollow pin−hole). Similarly, some pins on the sensor might be missing. A sensor can be connected to a socket if all pins could be plugged into corresponding pin−holes.


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
A socket with pin-holes #3, #4, #11, #13, and #15 not-hollow
A sensor with 6 missing pins (#3, #4, #6, #11, #13, and #15)

For example, a socket with 5 pin−holes being not hollow and a sensor with 6 missing pins are in the figure. This sensor could be connected to the socket because for all 9 pins it has, the socket has corresponding hollow pin−holes (here, the correspondence is remarked by numbers from 1 to 15). Note that the pin #6 is missing in the sensor while the pin−hole #6 is hollow, but it can still be connected.

Given 3 sockets and 3 sensors, write a program to determine whether the three sensors can be connected to the three sockets. Each sensors could be connected to any socket, but all three sensors must be connected to distinct sockets simultaneously.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains two integers X and Y ($3 \le X, Y \le 10$), where Y is the number of pin−holes and pins in each row.

The next $3X$ lines describe the three sensors in binary representation, such that the first X lines for the first sensor, the following X lines for the second sensor, and the last X lines for the third sensor. Each row will contain a binary string of length Y where 0 indicates that the pin is missing and 1 indicates that there is a pin.

The next $3X$ lines describe the three sockets in the same manner. For sockets, 0 indicates that the hole is not hollow and 1 indicates that it is hollow.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, print ‘`YES`’(quotations for clarification only) if it is possible to connect all three sensors. Otherwise, print ‘`NO`’.

Sample input and output

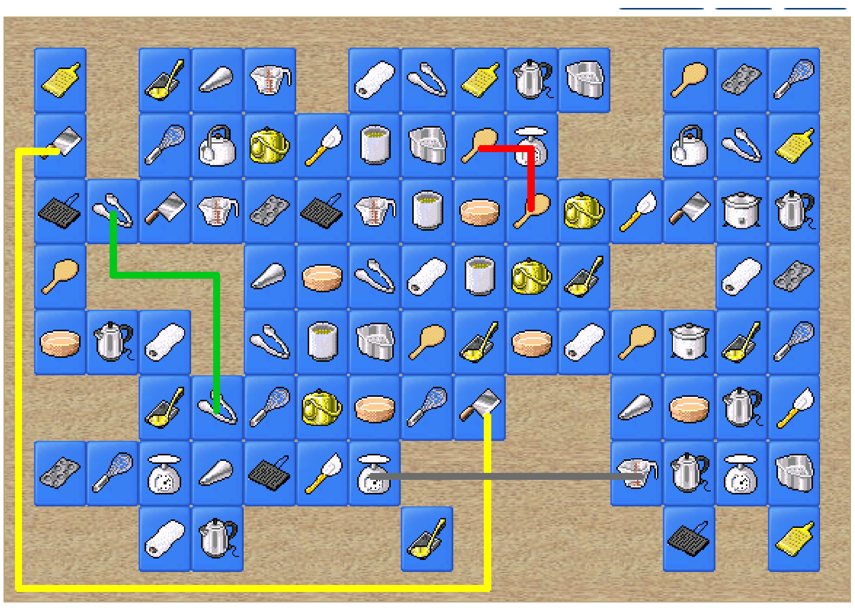
Standard Input	Standard Output
2	YES
3 5	NO
00000	
01010	
00110	
00000	
10101	
11111	
00111	
00000	
01001	
00000	
01110	
01110	
00011	
10111	
11111	
01111	
00000	
01111	
3 3	
000	
011	
000	
010	
010	
000	
000	
110	
000	
000	
010	
010	
000	
110	
000	
000	
110	
000	

Problem B. Shisen-sho

Shisen-sho(in Korean, *Sa-Cheon-Sung*) is a board game that originated in Japan, which is popular in Korea as well. The game is played on a rectangular game board with $N \times M$ cells, and various tiles are placed on some of the cells. The object of the game is to remove all of the tiles from the board, and the only way to remove tiles from the board is to remove one pair of matching tiles connected by a **valid path**. Throughout the game play, the player continues to remove pairs of tiles from the board, pair by pair.

A **valid path** is a path on the game board which satisfies the following:

- The path connects two matching tiles.
- The path consists of at most three axis-aligned line segments, starting and ending at the center of two tiles.
- The path should not touch or pass through any cells containing tiles, except the starting and ending tiles.



For example, the above figure shows a possible game board and some valid and invalid paths.

- The green path is a valid one.
- The gray path is invalid, because it connects two different tiles.
- The yellow path is invalid, because it contains 4 line segments.
- The red path is invalid, because it passes through a cell that is not empty.

We want to assign difficulty levels to a set of Shisen-sho game boards. The difficulty level is defined by the number of distinct pairs of tiles that can be removed from the board with a single move. A single move means removing a pair of tiles connected by a valid path. Given a game board, Write a program to compute the difficulty level of the board.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains two integers H and W $(3 \le H, W \le 50)$. H lines will follow, each of which contains W characters where each character denotes a cell in the game board. A period (.) denotes an empty cell, and lower and upper alphabet characters denote different kind of tiles. For all test cases, the first row, first column, last row, and last column of the game board will be empty.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer indicating the number of distinct pair of tiles that can be removed from the board with a single move.

Sample input and output

Standard Input	Standard Output
2 4 5 ..... .A.A. ...A. ..... 4 7 .... .A.X.A. .A.Y.A.	3 4

Problem C. Minesweeper

`c0ffee`, a famous hacker, plays Minesweeper game a lot in his spare time. After hours and hours of playing Minesweeper, `c0ffee` finally got bored and wrote a hack program for the game. With this hack, `c0ffee` instantly knows which cells contain mines. `c0ffee` now wants to set a speed record using this hack. But alas, his mouse is broken so he cannot make right clicks. So he must clear the board using left clicks only.

In case you are rusty with Minesweeper, here is a quick refresher on how the game works:

- An empty cell is **opened** when you left click on it.
- When a cell is opened, it will show the number of mines in adjacent cells. Two cells are adjacent if they share an edge or a corner. See figure 1 for an exmple.
- If all the adjacent cells are empty(not containing a mine), all adjacent cells are opened automatically, and this process goes on until no more cells are automatically opened. Figure 2 shows an example; when the cell with X is clicked, all those cells will open at once.

Given a minsweeper board, write a program to compute the minimum number of left clicks `c0ffee` has to make in order to clear the given board.

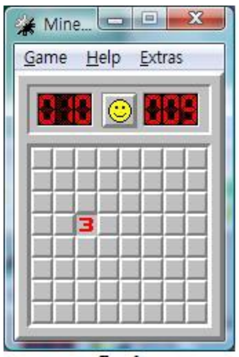


fig 1

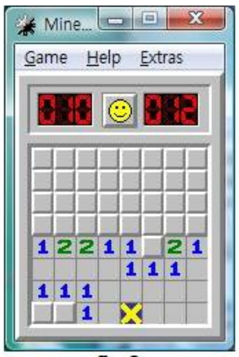


fig 2

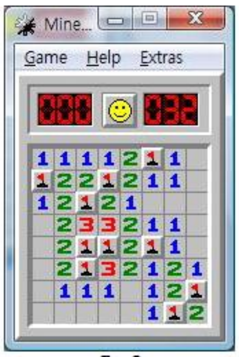


fig 3

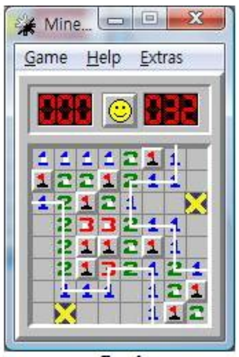


fig 4

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains two integers R and C ($1 \le R, C \le 200$), where R is the number of rows and C is the number of columns. The following R lines will each contain C characters, where mines are denoted by an asterisk ('*') and empty cells by a dot ('.').

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer indicating the minimum number of left clicks required to clear the board.

Notes

- Figure 3 describes the second sample input.

- Figure 4 describes the minimum clicks required to clear the board: cells marked with Xs and white dots must be clicked.

Sample input and output

Standard Input	Standard Output
2 3 4 .... *..* .... 8 8 .....*.. *..*.... ..*..... ....**.*.. ..*..... ....* ......*.	10 15

Problem D. What-the-Mole

Wonha works for a game development company nowadays. He had to learn several technologies, but he was tired of reading technical documents. So he decided to write and demonstrate a simple game. The name is chosen as **"What–the–Mole"** (due to copyright issues).

What–the–Mole contains a game panel that simply consists of nine holes that are aligned as a 3×3 grid. N moles appear on the holes randomly in position and time, and the objective of the game is to hit these moles as many times as possible.

A player uses a cursor to hit the moles. At the start of the game, cursor is on the top–left hole. There are two actions that the player may do. One action is to move the cursor to its adjacent holes (two holes are adjacent if they share an edge on the grid), and it takes 0.9 seconds. Note that the player cannot do anything else with the cursor while moving.

Another action is to swing the hammer on the current hole where the cursor is on. It's a little bit com–plicated as some animations and effects should be played. If a player tries to hit the mole, animations are played for 0.1 seconds. After that, the judgment of whether the hitting is successful or not takes place; if there is a mole, then the hitting is judged as successful, otherwise not. Successful hitting scores one point, and the hit mole will disappear immediately. If a mole appears exactly at the same time of the judgment, it will be judged as a successful hitting.

The moles will appear after a non–negative integer number of seconds after the start of the game, and will disappear 2.75 seconds after the appearance. Let us denote the duration of the appearance of the i–th mole as T_i, and the coordinates of the hole at which the i–th mole appears as (R_i, C_i). The mole will appear on the hole located at R_i–th row and C_i–th column, at time T_i.

Wonha was worried about the case where two or more moles appear and occupy the same hole – notice that even if two moles appear at different times, they can still occupy the same hole if their appearance time intervals overlap. Thus, he made some tricks in his program to avoid such cases — even if a player does not hit any moles, it is guaranteed that, at any point in time, no two moles will occupy the same hole at the same time. The game will end when the last mole disappears.

Wonha tested the game several times, and he found that his pseudo–random number generator is broken. If he changes the system time to specific moment and clicks the menu button several times, the pattern of the game will always be the same, just like Pokemon's RNG trick. Now he is curious about the maximum score for this specific pattern of the game. Help him to find out maximum score, given how the moles will appear and disappear.

Input

Your program is to read from standard input. The input constsist of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains an integer, N. Each line of the next N lines contains three integers R_i, C_i ($1 \le R_i, C_i \le 3$), and T_i ($0 \le T_i \le 100$), separated by space(s). Each line describes when and where a mole appears.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain a single integer: the maximum score for a given game pattern.

Sample input and output

Standard Input	Standard Output
1	3
4	
1 1 0	
2 2 1	
3 3 0	
3 3 3	

Problem E. Gandhi The Conqueror

Gandhi, one of the most famous conquerors and emperors, wants to conquer a new continent. This continent is rectangular, which is split into $H \times W$ square **regions** (think of a grid). Each region in this continent have a **policy**, which describes the culture of that region.

The regions of this continent work in a curious way, as follows:

- In the beginning, each region establishes a **nation** of its own.
- Whenever two nations share an edge and they have the same policy, they unite into a single nation.

Gandhi is the leader of the region located at the left-top corner of the continent. As Gandhi is getting old, he doesn't want violence anymore and wants to unite the whole continent in a peaceful way. His plan is to change his nation's policy every year, so he can unite with the neighboring nations. By this way, he only has to assassinate the leaders of other nations to conquer the whole continent.

Of course, changing a nation's policy takes a lot of time; and Gandhi doesn't want to do that many times. So he asks the Great Coder `ainu7` to minimize the number of changes of the policy.

순순히 마을을 넘기면 유혈사태는 일어나지 않을 것입니다.
I heartily thank you for the donation in the interests of world peace and tranquility.

Given a map of the continent, and the policy of each region, find the minimum number of times Gandhi has to change his nation's policy to unite the whole continent. It doesn't matter what policy Gandhi's nation ends up with.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains two integers H and W, which denote the height and the width of the continent, respectively ($H + W \le 30$). H lines will follow, each containing W integers between 0 and 4, inclusive. These numbers denote the current policy of each region.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the number of times Gandhi has to change his nation's policy.

Sample input and output

Standard Input	Standard Output

3
1 15
0 1 2 3 4 0 1 2 3 4 0 1 2 3 4
3 3
1 2 1
2 1 1
1 1 1
3 3
0 1 0
1 0 0
0 0 1
	14
	2
	3

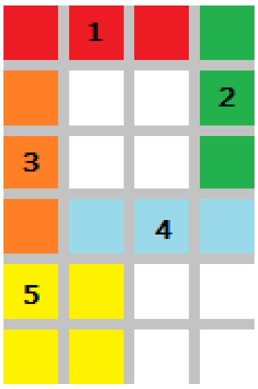
Problem F. A New Layout Algorithm

Lately, Eunjin is working hard on developing an application that, given a keyword, displays images gath–ered from the Internet to give the user some insight about the keyword.

Assuming that Eunjin already collected total N rectangular bitmap images of various sizes to display, she has to locate the images in a rectangular view panel that is W pixels wide. The panel will automatically span vertically so that it can fit all images. She uses the following rules to layout the images:

- As a rule of thumb, the images be considered sequentially, according to the given order.
- No image can be rotated.
- No two or more images shall be overlapped.
- For each column of the view panel, the image indices of the non–empty pixels should be in non–decreasing order, when taken from top to bottom. That is, an image with higher index should not be placed above another image with lower index.
- Among all of the feasible positions of an image, the algorithm must choose the top–most position (for its left–top pixel); if there are ties, choose the left–most one (for its left–top pixel).

The following figure illustrates the sample input. The numbers represent the images. Note that fifth image cannot placed in the 2 by 2 empty pixels surrounded by other four images as doing so will violate the rules above.



Compute the required minimum height of the view panel, given a set of images.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains two integers N $(1 \leq N \leq 2,000)$ and W $(1 \leq W \leq 1,920)$ separated by whitespace(s). Each line of the next N lines will contain two integers X_i $(1 \leq X_i \leq W)$ and Y_i $(1 \leq Y_i \leq 1,200)$, indicating that the i–th image is X_i pixels wide and Y_i pixels high.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the minimum height of the view panel required for displaying the given images.

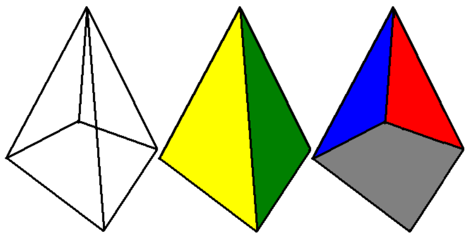
Sample input and output

Standard Input	Standard Output

1
5 4
3 1
1 3
1 3
3 1
2 2
	6

Problem G. Visibility Measure

In computer graphics, identifying visible lines and surfaces is an important task. When a polygon or a polyhedron is rendered by a graphical program, only visible parts(lines and surfaces) are considered and rendered. This might reduce the computational time and space, and most of all, it makes the rendered object to be realistic.



Consider a polyhedron(pentahedron) in the figure. When it is viewed from the front, only two surfaces are visible and rendered as in the figure in the middle. However, if the other surfaces are rendered as in the right−most figure, the result is not realistic(it is rather like bottom−back view).

When a polygon on two dimensional plane is given, the visibility measure could be defined roughly as follows. If there exists a point outside the polygon where the number of visible line segments of the polygon is K, we say that the polygon is K−visible. The **visibility measure** of the polygon is the maximum value of K.

A formal definition is as follows: First, let S be the set of points on the boundary of a polygon. For a point $q \notin S$, a point $p \in S$ is visible from q if and only if

$$\nexists r \in S \text{ such that } p \neq r \ \wedge \ p,\ q,\ r \text{ are co−linear } \wedge \ |\overline{qr}| < |\overline{qp}|$$

where $|\overline{qr}|$ denotes the distance between q and r. For a fixed q, we define $V(q, S)$ as the set of line segments of polygon S which are visible from q. If there exists a point $q' \notin S$ where the cardinality(number of elements) of $V(q', S)$ is K, we say that S is K−visible. The **visibility measure** of S is the maximum value of K which satisfies that S is K−visible.

Given a convex polygon S on two dimensional plane, write a program to compute the visibility measure of S.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains an integer, n ($3 \leq n \leq 20,000$), the number of vertices of S. Each line of the next n lines contains two integers, x and y, where (x, y) is the coordinates of each vertex ($-10^9 \leq x, y \leq 10^9$). For each test case, the order of vertices would be either clockwise or counter−clockwise, and no three points (vertices) are co−linear.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer indicating the visibility measure of the given polygon.

Sample input and output

Standard Input	Standard Output

2
4
-2 1
2 1
2 -1
-2 -1
7
0 0
127 0
205 69
180 153
97 169
13 152
-68 76
	2
	4

Problem H. Typesetting Deadline

The administrators of **algospot.com** are busy preparing yet another programming contest. They have selected N problems from the problem pool that will appear in the next contest.

Unfortunately, at this time, only one of the administrators, *the* L*ord of the* T*remendously* D*azzling and* T*errific* L*and*, is able to typeset the problems. Because typesetting requires a lot of concentration, the administrator may process one problem at a time. Yet she may freely choose the order of the problems to typeset. Each problem requires a different amount of time of typesetting. Let us denote the required typesetting time for the i−th problem as T_i.

The director of the contest is going to set a **deadline** for typesetting. As a result, the typesetter, **LTDTL** will feel some sort of **uneasiness**. Let us consider the following model for measuring uneasiness, given the deadline D:

$$\sum_{i\in[1,N]} A\cdot D + B\cdot \mathrm{Tardiness}_i + C\cdot \mathrm{Earliness}_i$$

where A, B and C are non−negative integer constants.

Before defining Tardiness and Earliness, let us take a close look at the first term. $A\cdot D$ represents the psychological pressure due to the late deadline. Although it appears to the typesetter that she has more time to typeset the problems, it also means that the typesetter will be occupied by all the hectic work for a longer period of time, and thus she will be feeling uneasy for a long time. Hence, the further the deadline is from now, the more uneasy the typesetter will feel.

$\mathrm{Tardiness}_i$ means the 'lateness' of the i−th problem. $\mathrm{Tardiness}_i$ is zero if the typesetting for the i−th problem will be done no later than the deadline. Otherwise, it will be defined as the difference between the finish time of the i−th problem and the deadline.

Analogously, $\mathrm{Earliness}_i$ is zero if the typesetting of the i−th problem is done later than the deadline. Otherwise, it will be defined as the difference between the finish time of the i−th problem and the deadline.

For example, if we assume $D = 5$ and the finish time of the problem is 7, then the uneasiness value for that problem will be $A\cdot 5 + B\cdot 2 + C\cdot 0$.

The minimum amount of the uneasiness will be dependent on the deadline and the order of typesetting problems. Write a program that finds the minimum amount of uneasiness.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains four integers N $(1 \le N \le 1,000)$, A, B, and C $(0 \le A, B, C \le 100)$. The next line will contain N integers, T_i $(1 \le T_i \le 1,000)$, separated by spaces.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the minimum uneasiness.

Sample input and output

Standard Input	Standard Output

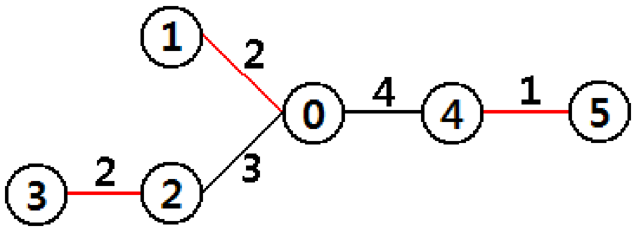
2
2 2 1 3
2 2
2 1 3 2
3 1
	6
	9

Problem I. Snow-Plow Car

Seoul National University (SNU), colloquially known in Korean as **Seoul–dae**, is a national research university in Seoul, Korea. Originally, the main campus, which embraced the College of Humanities and Sciences and College of Law, was located on **Daehangno (University Street)** in **Jongno**. Most parts of the university were relocated to a new campus in **Gwanak** in the period between 1975 and 1979, where Gwanak campus is located on the northwest slope of **Gwanaksan**, a small mountain in Seoul.

Last winter, it snowed often in Korea, and particularly, it snowed heavily in **Gwanak–gu**, a district of Seoul where SNU and Gwanaksan are located. When it snows heavily, cars and buses cannot move along the roads which are on the slope of Gwanaksan. Lots of building are located all around the campus, and the buildings are connected with a bidirectional roads. Since there is exactly one simple path(a path without any duplicate road) from one building to another, professors and students claim for their inconvenience every snowy winter.

The administration center of SNU has bought snow–plow cars(a vehicle intended for removing snow from the road) and placed them on every road. However, as it snowed heavily last winter, all snow–plow cars are now broken. The administration center decided to fix snow–plow cars, but due to limited budget, they cannot fix every single snow–plow car. Therefore, they decided to pick some of the snow–plow cars to be fixed, in order to make it easy to remove snow from the roads on the mountain slope (note that each road may or may not be on the mountain slope). For each slope road, the snow on the road can be removed if the snow–plow car on the road itself is fixed or that on any of its adjacent road is fixed. Two roads are said to be adjacent if they are connected to the same building.



For example, on the above figure, there are six building numbered from 0 to 5, and there are 3 roads in red (namely, $0-1$, $2-3$, and $4-5$) that are on the mountain slope. The number on the road is the cost to fix the snow–plow car on that road. If all snow–plow cars on slope roads are fixed, it costs $5(=2+2+1)$. On the other hand, if snow–plow cars on road $0-4$ and $2-3$ are fixed, it costs $6 = (4+2)$. snow–plow car on road $0-4$ covers two slope roads, $0-1$ and $4-5$. The optimal way is to fix the snow–plow cars on road $0-2$, $4-5$, which costs $4(=3+1)$. In this case, the snow–plow car on road $0-2$ covers road $0-1$ and $2-3$.

Given a tree representing the buildings and roads of SNU, write a program to compute the minimum cost to fix snow–plow cars in order to satisfy the requirements.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains an integer, n ($1 \leq n \leq 10,000$), the number of buildings in SNU. Each line of the next $n-1$ lines contains four integers, x, y, z, and c, where there is road connecting building number x and y. Third integer z will be 1 if the road is on the mountain slope, and 0 if the road is **not** on the mountain slope. The cost for fixing a snow−plow car on this road is $c(1 \leq c \leq 100,000)$ Building numbers will be distinct integers from 0 to $n-1$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the minimum cost to fix snow−plow cars.

Sample input and output

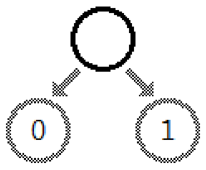
Standard Input	Standard Output

3
6
0 1 1 2
3 2 1 2
2 0 0 3
4 0 0 4
4 5 1 1
4
0 1 1 1
1 2 1 2
2 3 1 4
6
0 4 1 2
4 5 0 3
5 2 1 2
3 5 1 2
4 1 1 2
	4
	2
	3

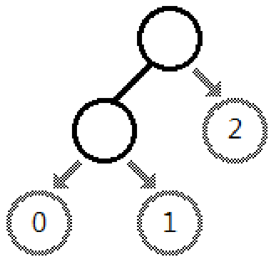
Problem J. Tree Code

Taeyoon, a professional computer programmer, has made a problem related to the binary tree structure in the latest contest. After writing the problem description and the judge solution, he had to make test cases which would be used in the contest. However, making various tree structure was not an easy problem. After long and thoughtful consideration, he finally came up with an idea called the **tree code**.

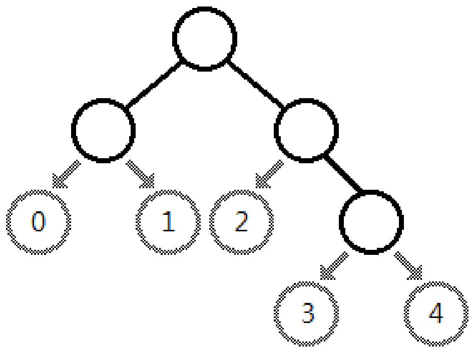
Consider a simple binary tree with a single (root) node. Then, we can choose one of the two branches in order to add a node.



After adding a child node to the root, now the tree has two nodes. Then again, we can choose one of the three branches to add another node.



Put differently, if we want to add a new node to the current binary tree with K nodes, we should choose one of the $K+1$ branches. As we can see from the figure below, these $K+1$ branches can be labeled using numbers between 0 and K, inclusive, according to their appearance in the in−order search performed on the tree (informally, the left−most node will be labeled as 0 as in the figure below).



As we build a binary tree from a single root node, if we record the sequence of the branch numbers to which we add a new node, then the resulted string can describe a binary tree we end up with. This string is called the *tree code* for the binary tree we construct.

A tree code consists of numbers and uppercase alphabets. Branch numbers 0 to 9 are described as '`0`' to '`9`', and 10 to 35 as '`A`' to '`Z`'. One tree code can uniquely describe a binary tree, but a tree can be described by a number of different tree codes. For example, both `02` and `10` describe the binary tree consisting of a root and its two children.

You are given a binary tree and asked to find the lexicographically M−th tree code that describes the given tree.

A tree code A is lexicographically smaller than a tree code B of the same length if the i−th character of A is smaller than the i−th character of B where i is the first position where A and B differ ('`0`' is smaller than '`1`'). Character−wise, a numeric digit (0 – 9) is lexicographically smaller than any alphabet.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains a tree code describing a binary tree, and an integer $M (1 \leq M \leq 10^{18})$. Each tree code will contain at least one character, and its length will not exceed 36. You can assume that all tree codes given in the input data are valid.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the lexicographically M−th tree code which can describe the given tree. If the number of tree codes that describe the given tree is smaller than M, print '`IMPOSSIBLE`'(without quotation marks).

Sample input and output

Standard Input	Standard Output

2
02 2
1010 4
	10
	0130