Codeforces Round #644 (Div. 3)

A. Minimal Square

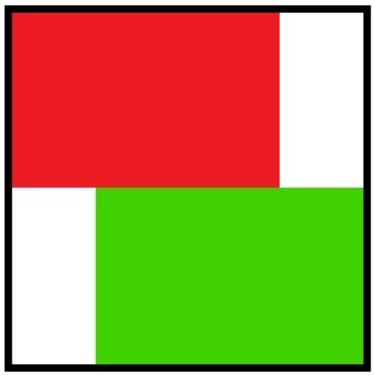
2 seconds, 256 megabytes

Find the minimum area of a **square** land on which you can place two identical rectangular $a \times b$ houses. The sides of the houses should be parallel to the sides of the desired square land.

Formally,

- You are given two identical rectangles with side lengths a and b ($1 \le a, b \le 100$) — positive integers (you are given just the sizes, but **not** their positions).
- Find the square of the minimum area that contains both given rectangles. Rectangles can be rotated (both or just one), moved, but the sides of the rectangles should be parallel to the sides of the desired square.

Two rectangles can touch each other (side or corner), but cannot intersect. Rectangles can also touch the sides of the square but must be completely inside it. You can rotate the rectangles. Take a look at the examples for a better understanding.



The picture shows a square that contains red and green rectangles.

Input

The first line contains an integer t ($1 \le t \le 10\,000$) —the number of test cases in the input. Then t test cases follow.

Each test case is a line containing two integers a, b ($1 \le a, b \le 100$) — side lengths of the rectangles.

Output

Print t answers to the test cases. Each answer must be a single integer — minimal area of square land, that contains two rectangles with dimensions $a \times b$.

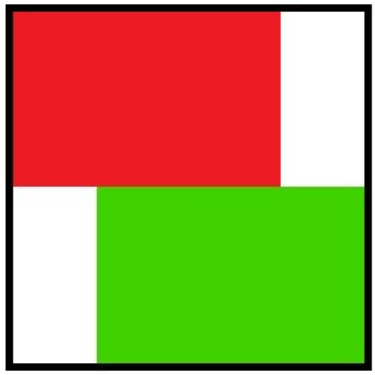
input

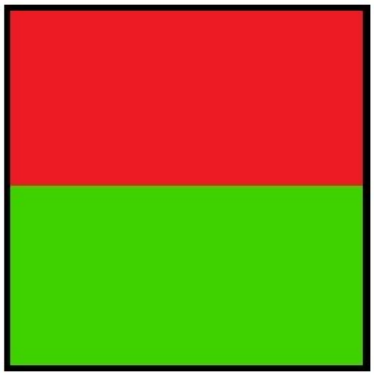
```
8
3 2
4 2
1 1
3 1
4 7
1 3
7 4
100 100
```

output

```
16
16
4
9
64
9
64
40000
```

Below are the answers for the first two test cases:





B. Honest Coach

2 seconds, 256 megabytes

There are n athletes in front of you. Athletes are numbered from 1 to n from left to right. You know the strength of each athlete — the athlete number i has the strength s_i.

You want to split all athletes into two teams. Each team must have at least one athlete, and each athlete must be exactly in one team.

You want the strongest athlete from the first team to differ as little as possible from the weakest athlete from the second team. Formally, you want to split the athletes into two teams A and B so that the value $|\max(A) - \min(B)|$ is as small as possible, where $\max(A)$ is the maximum strength of an athlete from team A, and $\min(B)$ is the minimum strength of an athlete from team B.

For example, if $n = 5$ and the strength of the athletes is $s = [3, 1, 2, 6, 4]$, then one of the possible split into teams is:

- first team: $A = [1, 2, 4]$,
- second team: $B = [3, 6]$.

In this case, the value $|\max(A) - \min(B)|$ will be equal to $|4 - 3| = 1$. This example illustrates one of the ways of optimal split into two teams.

Print the minimum value $|\max(A) - \min(B)|$.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases in the input. Then t test cases follow.

Each test case consists of two lines.

The first line contains positive integer n ($2 \le n \le 50$) — number of athletes.

The second line contains n positive integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 1000$), where s_i — is the strength of the i-th athlete. Please note that s values may not be distinct.

Output

For each test case print one integer — the minimum value of $|\max(A) - \min(B)|$ with the optimal split of all athletes into two teams. Each of the athletes must be a member of exactly one of the two teams.

input

```
5
5
3 1 2 6 4
6
2 1 3 2 4 3
4
7 9 3 1
2
1 1000
3
100 150 200
```

output

```
1
0
2
999
50
```

The first test case was explained in the statement. In the second test case, one of the optimal splits is $A = [2, 1]$, $B = [3, 2, 4, 3]$, so the answer is $|2 - 2| = 0$.

C. Similar Pairs

2 seconds, 256 megabytes

We call two numbers x and y *similar* if they have the same parity (the same remainder when divided by 2), or if $|x - y| = 1$. For example, in each of the pairs $(2, 6)$, $(4, 3)$, $(11, 7)$, the numbers are similar to each other, and in the pairs $(1, 4)$, $(3, 12)$, they are not.

You are given an array a of n (n is even) positive integers. Check if there is such a partition of the array into pairs that each element of the array belongs to exactly one pair and the numbers in each pair are similar to each other.

For example, for the array $a = [11, 14, 16, 12]$, there is a partition into pairs $(11, 12)$ and $(14, 16)$. The numbers in the first pair are similar because they differ by one, and in the second pair because they are both even.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

Each test case consists of two lines.

The first line contains an **even** positive integer n ($2 \le n \le 50$) — length of array a.

The second line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$).

Output

For each test case print:

- `YES` if the such a partition exists,
- `NO` otherwise.

The letters in the words `YES` and `NO` can be displayed in any case.

input

```
7
4
11 14 16 12
2
1 8
4
1 1 1 1
4
1 2 5 6
2
12 13
6
1 6 3 10 5 8
6
1 12 3 10 5 8
```

output

```
YES
NO
YES
YES
YES
YES
NO
```

The first test case was explained in the statement.

In the second test case, the two given numbers are not similar.

In the third test case, any partition is suitable.

D. Buying Shovels

2 seconds, 256 megabytes

Polycarp wants to buy **exactly** n shovels. The shop sells packages with shovels. The store has k types of packages: the package of the i-th type consists of exactly i shovels ($1 \le i \le k$). The store has an infinite number of packages of each type.

Polycarp wants to choose **one** type of packages and then buy several (one or more) packages of this type. What is the smallest number of packages Polycarp will have to buy to get exactly n shovels?

For example, if $n = 8$ and $k = 7$, then Polycarp will buy 2 packages of 4 shovels.

Help Polycarp find the minimum number of packages that he needs to buy, given that he:

- will buy exactly n shovels in total;
- the sizes of **all** packages he will buy are all the same and the number of shovels in each package is an integer from 1 to k, inclusive.

Input
The first line contains an integer t ($1 \le t \le 100$) — the number of test cases in the input. Then, t test cases follow, one per line.

Each test case consists of two positive integers n ($1 \le n \le 10^9$) and k ($1 \le k \le 10^9$) — the number of shovels and the number of types of packages.

Output
Print t answers to the test cases. Each answer is a positive integer — the minimum number of packages.

input

```
5
8 7
8 1
6 10
999999733 999999732
999999733 999999733
```

output

```
2
8
1
999999733
1
```

The answer to the first test case was explained in the statement.

In the second test case, there is only one way to buy 8 shovels — 8 packages of one shovel.

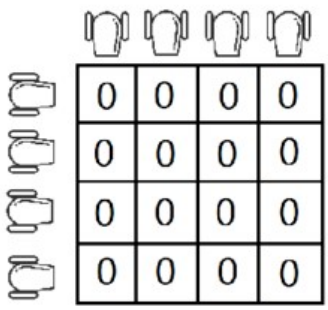
In the third test case, you need to buy a 1 package of 6 shovels.

E. Polygon

2 seconds, 256 megabytes

Polygon is not only the best platform for developing problems but also a square matrix with side n, initially filled with the character `0`.

On the polygon, military training was held. The soldiers placed a cannon above each cell in the first row and a cannon to the left of each cell in the first column. Thus, exactly $2n$ cannons were placed.



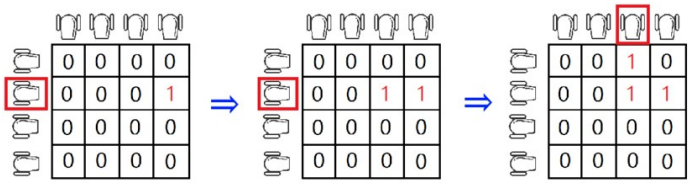
Initial polygon for $n = 4$.

Cannons shoot character `1`. At any moment of time, no more than one cannon is shooting. When a `1` flies out of a cannon, it flies forward (in the direction of the shot) until it collides with a polygon border or another `1`. After that, it takes the cell in which it was before the collision and remains there. Take a look at the examples for better understanding.

More formally:

- if a cannon stands in the row i, to the left of the first column, and shoots with a `1`, then the `1` starts its flight from the cell $(i, 1)$ and ends in some cell (i, j);
- if a cannon stands in the column j, above the first row, and shoots with a `1`, then the `1` starts its flight from the cell $(1, j)$ and ends in some cell (i, j).

For example, consider the following sequence of shots:



1. Shoot the cannon in the row 2.
2. Shoot the cannon in the row 2.
3. Shoot the cannon in column 3.

You have a report from the military training on your desk. This report is a square matrix with side length n consisting of `0` and `1`. You wonder if the training actually happened. In other words, is there a sequence of shots such that, after the training, you get the given matrix?

Each cannon can make an arbitrary number of shots. Before the training, each cell of the polygon contains `0`.

Input
The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

Each test case starts with a line containing an integer n ($1 \le n \le 50$) — the size of the polygon.

This is followed by n lines of length n, consisting of `0` and `1` — the polygon matrix after the training.

The total area of the matrices in all test cases in one test does not exceed 10^5.

Output
For each test case print:

- `YES` if there is a sequence of shots leading to a given matrix;
- `NO` if such a sequence does not exist.

The letters in the words `YES` and `NO` can be printed in any case.

input

```
5
4
0010
0011
0000
0000
2
10
01
2
00
00
4
0101
1111
0101
0111
4
0100
1110
0101
0111
```

output

```
YES
NO
YES
YES
NO
```

The first test case was explained in the statement.

The answer to the second test case is `NO`, since a `1` in a cell $(1, 1)$ flying out of any cannon would continue its flight further.

F. Spy-string

2 seconds, 256 megabytes

You are given n strings $a_1, a_2, \ldots, a_n$: all of them have the same length m. The strings consist of lowercase English letters.

Find any string s of length m such that each of the given n strings differs from s in at most one position. Formally, for each given string a_i, there is no more than one position j such that $a_i[j] \neq s[j]$.

Note that the desired string s may be equal to one of the given strings a_i, or it may differ from all the given strings.

For example, if you have the strings `abac` and `zbab`, then the answer to the problem might be the string `abab`, which differs from the first only by the last character, and from the second only by the first.

Input

The first line contains an integer t ($1 \leq t \leq 100$) — the number of test cases. Then t test cases follow.

Each test case starts with a line containing two positive integers n ($1 \leq n \leq 10$) and m ($1 \leq m \leq 10$) — the number of strings and their length.

Then follow n strings a_i, one per line. Each of them has length m and consists of lowercase English letters.

Output

Print t answers to the test cases. Each answer (if it exists) is a string of length m consisting of lowercase English letters. If there are several answers, print any of them. If the answer does not exist, print "`-1`" ("minus one", without quotes).

input

```
5
2 4
abac
zbab
2 4
aaaa
bbbb
3 3
baa
aaa
aab
2 2
ab
bb
3 1
a
b
c
```

output

```
abab
-1
aaa
ab
z
```

The first test case was explained in the statement.

In the second test case, the answer does not exist.

G. A/B Matrix

2 seconds, 256 megabytes

You are given four positive integers n, m, a, b ($1 \leq b \leq n \leq 50$; $1 \leq a \leq m \leq 50$). Find any such rectangular matrix of size $n \times m$ that satisfies all of the following conditions:

- each row of the matrix contains exactly a ones;
- each column of the matrix contains exactly b ones;
- all other elements are zeros.

If the desired matrix does not exist, indicate this.

For example, for $n = 3$, $m = 6$, $a = 2$, $b = 1$, there exists a matrix satisfying the conditions above:

$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix}$$

Input

The first line contains an integer t ($1 \leq t \leq 1000$) — the number of test cases. Then t test cases follow.

Each test case is described by four positive integers n, m, a, b ($1 \leq b \leq n \leq 50$; $1 \leq a \leq m \leq 50$), where n and m are the sizes of the matrix, and a and b are the number of ones for rows and columns, respectively.

Output

For each test case print:

- "YES" (without quotes) and the required matrix (if there are several answers, print any) if it exists, or
- "NO" (without quotes) if it does not exist.

To print the matrix $n \times m$, print n rows, each of which consists of m numbers 0 or 1 describing a row of the matrix. Numbers must be printed **without spaces**.

input

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

output

```
YES
010001
100100
001010
NO
YES
11
11
YES
1100
1100
0011
0011
YES
1
1
```

H. Binary Median

2 seconds, 256 megabytes

Consider all binary strings of length m ($1 \le m \le 60$). A binary string is a string that consists of the characters `0` and `1` only. For example, `0110` is a binary string, and `012aba` is not. Obviously, there are exactly 2^m such strings in total.

The string s is lexicographically smaller than the string t (both have the same length m) if in the first position i from the left in which they differ, we have $s[i] < t[i]$. This is exactly the way strings are compared in dictionaries and in most modern programming languages when comparing them in a standard way. For example, the string `01011` is lexicographically smaller than the string `01100`, because the first two characters are the same, and the third character in the first string is less than that in the second.

We remove from this set n ($1 \le n \le \min(2^m - 1, 100)$) **distinct** binary strings $a_1, a_2, \ldots, a_n$, each of length m. Thus, the set will have $k = 2^m - n$ strings. Sort all strings of the resulting set in lexicographical ascending order (as in the dictionary).

We number all the strings after sorting from 0 to $k - 1$. Print the string whose index is $\lfloor \frac{k-1}{2} \rfloor$ (such an element is called *median*), where $\lfloor x \rfloor$ is the rounding of the number down to the nearest integer.

For example, if $n = 3$, $m = 3$ and $a = [010, 111, 001]$, then after removing the strings a_i and sorting, the result will take the form: $[000, 011, 100, 101, 110]$. Thus, the desired median is 100.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. Then, t test cases follow.

The first line of each test case contains integers n ($1 \le n \le \min(2^m - 1, 100)$) and m ($1 \le m \le 60$), where n is the number of strings to remove, and m is the length of binary strings. The next n lines contain $a_1, a_2, \ldots, a_n$ — **distinct** binary strings of length m.

The total length of all given binary strings in all test cases in one test does not exceed 10^5.

Output

Print t answers to the test cases. For each test case, print a string of length m — the median of the sorted sequence of remaining strings in the corresponding test case.

input

```
5
3 3
010
001
111
4 3
000
111
100
011
1 1
1
1 1
0
3 2
00
01
10
```

output

```
100
010
0
1
11
```

The first test case is explained in the statement.

In the second test case, the result after removing strings and sorting is $[001, 010, 101, 110]$. Therefore, the desired median is 010.