2023 Denison Spring Programming Contest
Granville, Ohio
11 February, 2023

Rules:

1. There are **six** problems to be completed in **four hours**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output.
3. No whitespace should appear in the output except between printed fields.
4. All whitespace, either in input or output, will consist of exactly one blank character.
5. The allowed programming languages are all available on Kattis (including C, C++, Python, and Java).
6. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required. However, the more readable your code, the more likely judges will be to give you a hint if you are stuck.
7. All communication with the judges will be handled email to lalla@denison.edu.
8. No cheating will be tolerated.

Boss Battle

You are stuck at a boss level of your favourite video game. The boss battle happens in a circular room with n indestructible pillars arranged evenly around the room. The boss hides behind an unknown pillar. Then the two of you proceed in turns.

- First, in your turn, you can throw a bomb past one of the pillars. The bomb will defeat the boss if it is behind that pillar, or either of the adjacent pillars.
- Next, if the boss was not defeated, it may either stay where it is, or use its turn to move to a pillar that is adjacent to its current position. With the smoke of the explosion you cannot see this movement.

The last time you tried to beat the boss you failed because you ran out of bombs. This time you want to gather enough bombs to make sure that whatever the boss does you will be able to beat it. What is the minimum number of bombs you need in order to defeat the boss in the worst case? See Figure 1 for an example.

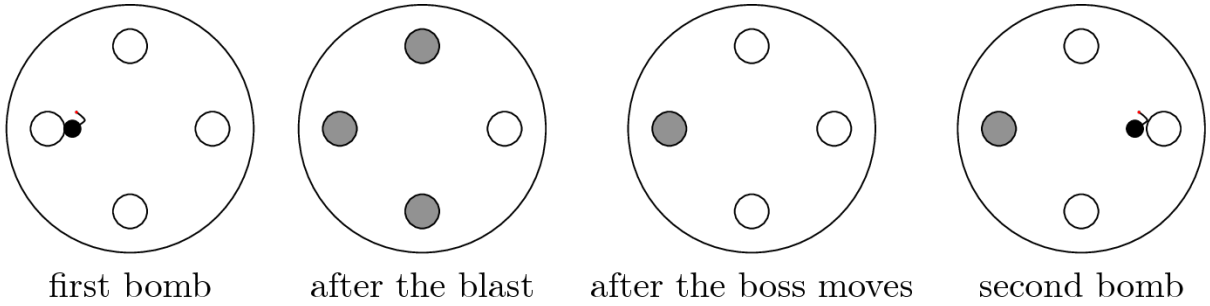


Figure 1: Example for n=4. In this case 2 bombs are enough. Grey pillars represent pillars where the boss cannot be hiding. The bomb is represented in black.

Input

The input consists of:

- One line with a single integer n (1≤n≤100), the number of pillars in the room.

Output

Output the minimum number of bombs needed to defeat the boss in the worst case.

Sample Input 1	Sample Output 1
4	2

Sample Input 2	Sample Output 2
7	5

Coin Stacks

Picture by KMR Photography via Wikimedia Commons, cc by

A and B are playing a collaborative game that involves n stacks of coins, numbered from 1 to n. Every round of the game, they select a nonempty stack each, but they are not allowed to choose the same stack. They then remove a coin from both the two selected stacks and then the next round begins.

The players win the game if they manage to remove all the coins. Is it possible for them to win the game, and if it is, how should they play?

Input

The first line of input contains an integer n ($2 \leq n \leq 50$), the number of coin stacks. Then follows a line containing n nonnegative integers $a_1, a_2, \ldots, a_n$, where a_i is the number of coins in the i'th stack. The total number of coins is at most 1000.

Output

If the players can win the game, output a line containing "`yes`", followed by a description of the moves. Otherwise output a line containing "`no`". When describing the moves, output one move per line, each move being described by two distinct integers a and b (between 1 and n) indicating that the players remove a coin from stacks a and b. If there are several possible solutions, output any one of them.

Sample Input 1

```
3
1 4 3
```

Sample Output 1

```
yes
1 2
2 3
2 3
2 3
```

Sample Input 2

```
3
1 1 1
```

Sample Output 2

```
no
```

Installing Apps

Mobile radio telephone, public domain

Sandra recently bought her first smart phone. One of her friends suggested a long list of applications (more commonly known as “apps”) that she should install on the phone. Sandra immediately started installing the apps from the list, but after installing a few, the phone did not have enough disk space to install any more apps. Sometimes, the app installation failed because there was not even enough space to download the installation package. Other apps could be downloaded just fine, but had insufficient space to store the installed app.

Each app that Sandra installs has a download size d and a storage size s. To download the app, Sandra's phone must have at least d megabytes of free disk space. After the app has been installed, it then uses s megabytes of disk space on the phone. The download size may be smaller than the storage size (e.g., if the app data is heavily compressed) or larger than the storage size (e.g., if the download contains material that might not get used such as translations to different languages). The installer is very efficient and can transform the downloaded package to an installed app without using any extra disk space. Thus, to install an app, the phone must have at least max(d,s) megabytes of free disk space.

Sandra quickly realised that she may have run out of space just because she installed apps in the wrong order. Thus, she decided to give the installation another try. She uninstalled all apps, and will now choose an installation order that lets her install the largest number of apps from the list. Sandra may not install any app more than once.

Help her determine what apps on the list she should install, and in what order.

Input

The input consists of:

- One line with two integers n, c ($1 \leq n \leq 500, 1 \leq c \leq 10000$), the number of available apps and the available disk space of the phone in megabytes.
- n lines, each with two integers d,s ($1 \leq d,s \leq 10000$), the download size and storage size of an app, in megabytes.

Output

Output one line with the maximum number of apps that can be installed. Then output one line listing the numbers of those apps, in the order that Sandra should install them. In the case that no apps can be installed, this line can be omitted.

The apps are numbered from 1 to n, in the order they are given in the input. If there are multiple optimal solutions, output any one of them.

Sample Input 1

```
2 100
99 1
1 99
```

Sample Output 1

```
2
1 2
```

Sample Input 2

```
2 100
500 1
1 500
```

Sample Output 2

```
0
```

Methodic Multiplication

Giuseppe Peano, public domain

After one computer crash too many, Alonso has had enough of all this shoddy software and poorly written code! He decides that in order for this situation to improve, the glass house that is modern programming needs to be torn down and rebuilt from scratch using only completely formal axiomatic reasoning. As one of the first steps, he decides to implement arithmetic with natural numbers using the Peano axioms.

The Peano axioms (named after Italian mathematican Giuseppe Peano) are an axiomatic formalization of the arithmetic properties of the natural numbers. We have two symbols: the constant 0, and a unary successor function S. The natural numbers, starting at 0, are then 0, S(0), S(S(0)), S(S(S(0))), and so on. With these two symbols, the operations of *addition* and *multiplication* are defined inductively by the following axioms: for any natural numbers x and y, we have

$$x+0=x \qquad x\cdot 0=0$$
$$x+S(y)=S(x+y) \qquad x\cdot S(y)=x\cdot y+x$$

The two axioms on the left define addition, and the two on the right define multiplication.

For instance, given $x=S(S(0))$ and $y=S(0)$ we can repeatedly apply these axioms to derive

$$x\cdot y=S(S(0))\cdot S(0)=S(S(0))\cdot 0+S(S(0))=0+S(S(0))=S(0+S(0))=S(S(0+0))=S(S(0))$$

Write a program which given two natural numbers x and y, defined in Peano arithmetic, computes the product $x\cdot y$.

Input

The input consists of two lines. Each line contains a natural number defined in Peano arithmatic, using at most 1000 characters.

Output

Output the product of the two input numbers.

Sample Input 1

```
S(S(0))
S(S(S(0)))
```

Sample Output 1

```
S(S(S(S(S(S(0))))))
```

Sample Input 2

```
S(S(S(S(S(0)))))
0
```

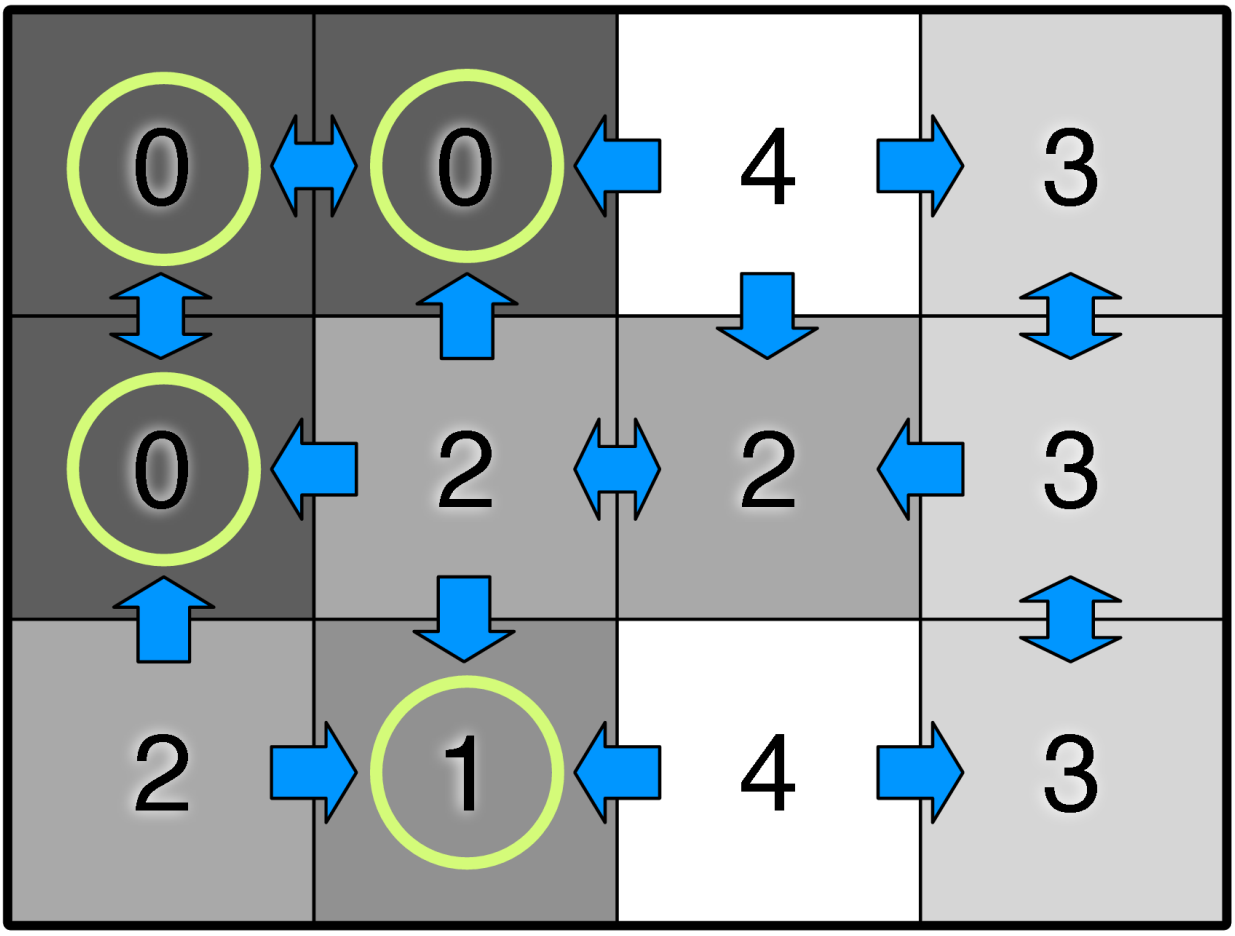
Sample Output 2

```
0
```

Terraces

After seeing all the beautiful landscaping around Cossin, Yraglac is inspired to work on his own gardening. Since food is such a scarce commodity on Mars, he decides it would be nice to plant an agricultural crop: Martian rice. The unique thing about Martian rice is that it will only grow in flat areas where water can pool, which makes it important to terrace the garden carefully. Given a height profile of Yraglac's garden, can you help him determine how much of the land can grow rice on?

Yraglac's garden is divided into a regular grid of perfectly square 1 m ×1 m cells, and he provides you a map indicating the exact height of the land within each cell. The entire garden is surrounded by a wall that is higher than the highest cell. In our model of the Mars world, rain water (real or artificial) can flow from a cell to any of its four adjacent cells (north, east, south, or west), provided the adjacent cell's height is lower than or equal to the cell's height. A cell is said to be able to collect or pool water if any water landing in the cell cannot flow to another cell of lower height, either directly or through any of its neighbouring cells. Arrows in the diagram below illustrate possible directions of water flow between cells, and circles indicate the four cells that can collect water.



Input

The input begins with two integers, xy, which indicate the dimensions of Yraglac's garden, in metres ($1 \leq x,y \leq 500$). Then following y lines containing x integers, hij, each which indicate the heights of each cell in Yraglac's garden ($0 \leq hij \leq 999$).

Output

Output the number of square metres of land that Yraglac can grow his rice crop on.

Sample Input 1	Sample Output 1

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

```
4
```

Sample Input 2

Sample Output 2

```
7 2
0 4 1 4 2 4 3
0 4 1 4 2 4 3
```

```
8
```

Sample Input 3

Sample Output 3

```
5 3
1 1 1 1 1
3 3 3 3 3
5 5 5 5 5
```

```
5
```

Sample Input 4

Sample Output 4

```
4 4
8 8 8 8
8 4 8 8
8 8 2 8
8 8 8 8
```

```
2
```

Unique Dice

A very large collection of dice.

You are about to leave home for your weekly game of Pumpkins and Flagons (P&F) when a text comes in asking you to bring a large collection of identical P&F dice. They have asked the right person, for you have a very large collection of P&F dice which, unfortunately, is currently unsorted.

These dice are ordinary cubes with a number on each of the six faces. The numbers are in the range 1–6, but they do not need to be distinct. For example, {1,2,3,4,5,6}, {1,1,1,1,1,1} and {2,2,2,4,4,5} are all valid sets of numbers for the six faces. For this problem you want to find the size of the largest set of identical dice you can create from your very large collection. Two dice are considered identical to each other if it is possible to rotate one of the

dice so that their top numbers are the same, their bottom numbers are the same, and so on for all six faces.

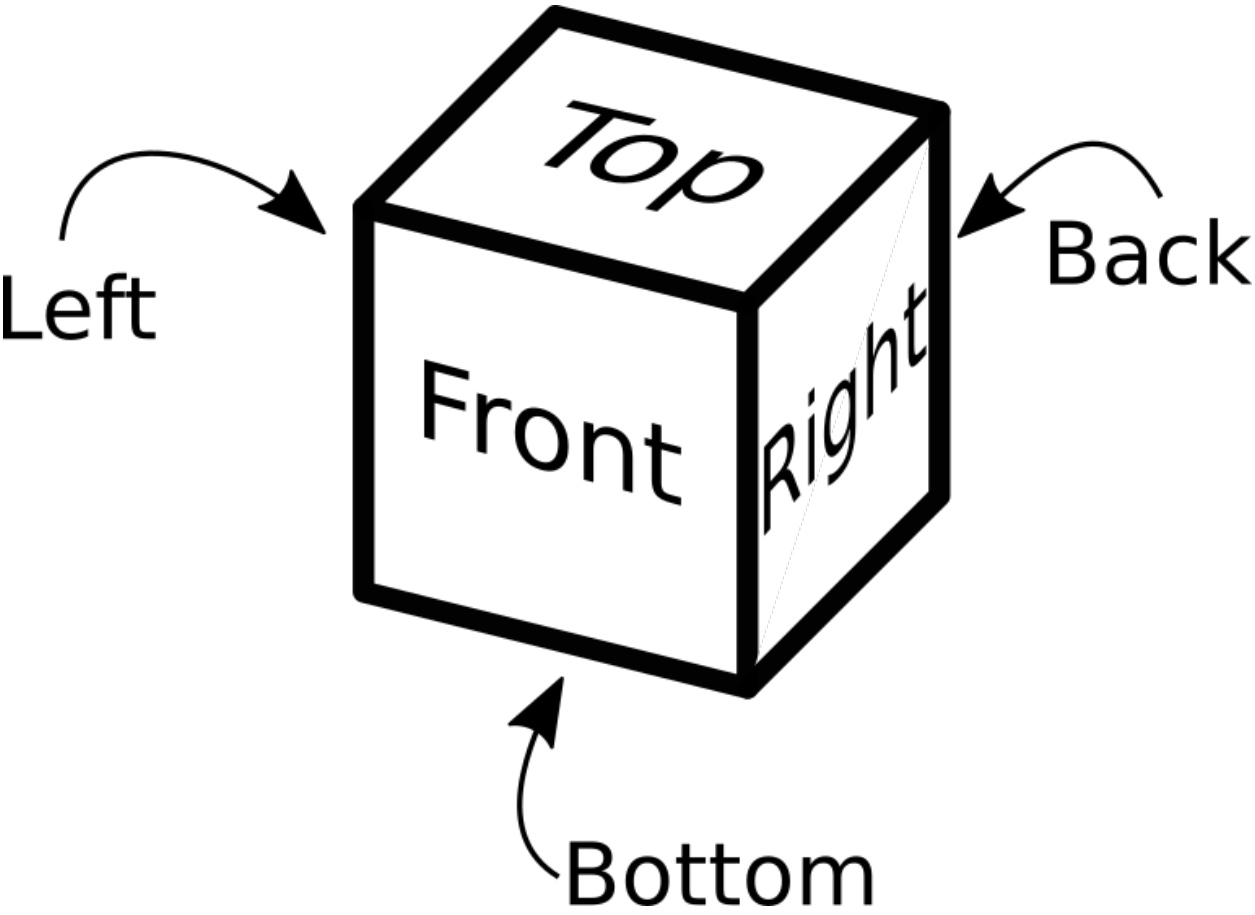


Figure 1: The six faces of a die.

Input

The first line contains an integer n ($1 \le n \le 500000$), indicating the number of dice in your collection. The next n lines each contains six integers in the range 1–6, separated by spaces, giving, in order, the numbers that appears on the top, bottom, front, back, left and right faces (see Figure 1).

Output

Print a single integer giving the size of the largest set of identical dice that can be made from given collection.

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
1
```

Sample Input 3

```
5
1 1 2 2 2 2
1 2 1 2 2 2
1 2 2 1 2 2
1 2 2 2 1 2
1 2 2 2 2 1
```

Sample Output 3

```
4
```