MIUP'21

Maratona Inter-Universitária de Programação

Instituto Superior Técnico - 9 de outubro de 2021

TÉCNICO LISBOA

Critical TechWorks | A BMW GROUP & CRITICAL SOFTWARE COMPANY

Unbabel

Maratona Inter–Universitária de Programação 2021

Teams from the following universities

CONTENTS

Scientific Committee

- André Restivo, FEUP / Universidade do Porto
- Fábio Marques, Universidade de Aveiro
- Filipe Araújo, Universidade de Coimbra
- Luís M. S. Russo, IST / Universidade de Lisboa
- Margarida Mamede, FCT / Universidade Nova de Lisboa
- Mário Pereira, FCT / Universidade Nova de Lisboa
- **Mikoláš Janota, Czech Technical University in Prague (Coordinator)**
- Pedro Guerreiro, Universidade do Algarve
- Pedro Mariano, ISCTE
- Pedro Ribeiro, FCUP / Universidade do Porto
- Rui Maranhão, FEUP / Universidade do Porto
- Rui Mendes, Universidade do Minho
- Simão Melo de Sousa, Universidade da Beira Interior
- Vasco Pedro, Universidade de Évora

Local Organization Committee

- Alexandre Francisco, IST / Universidade de Lisboa
- Arlindo Oliveira, IST / Universidade de Lisboa
- Francisco Santos, IST / Universidade de Lisboa
- Francisco Miguel Dionísio, IST / Universidade de Lisboa
- **João F. Ferreira, IST / Universidade de Lisboa (Coordinator)**
- José Carlos Monteiro, IST / Universidade de Lisboa
- Luís Guerra e Silva, IST / Universidade de Lisboa
- Pedro T. Monteiro, IST / Universidade de Lisboa
- Vasco Manquinho, IST / Universidade de Lisboa

Languages and Compilers

I. Files `*.c` are assumed to be C code and are compiled with `gcc 8.3.0` with the command

```
gcc -std=gnu17 -Wall name.c -lm
```

and then executed with the command

```
./a.out
```

II. Files `*.cpp` are assumed to be C++ code and are compiled with `gcc 8.3.0` with the command

```
g++ -std=gnu++17 -Wall name.cpp -lm
```

and then executed with the command

```
./a.out
```

III. Files `*.java` are assumed to be Java code and are compiled with `OpenJDK 11.0.12` with the command

```
javac -encoding utf8 name.java
```

and then executed with the command

```
java -Xmx256M -Xss32M -classpath .  name
```

IV. Files `*.py` are assumed to be Python 3 code and are executed with `Python 3.7.3` with the command

```
python3 name.py
```

Limits

Attention: The use of pragmas is explicitly prohibited in C/C++ solutions (such as modifying optimization level). Solutions containing pragmas will be considered invalid.

Compilation: The compilation process can take at most 60 seconds and the maximum source code size is 100 KB. Every program/solution must be submitted in a **single file**. For Java submissions, the file must have the same name as the class that contains the main method. There is no limit for the number of classes to be contained in that file.

Runtime Although the maximum runtime is defined individually for each problem, the overall limit is 2 seconds.

Input/Output

All programs should read the input from the standard input, and write the output to the standard output. All lines (both in the input and output) should end with the newline character (`\n`). Except when explicitly stated, single space is used as a separator. No line starts or ends with any kind of white space.

IDEs and Editors

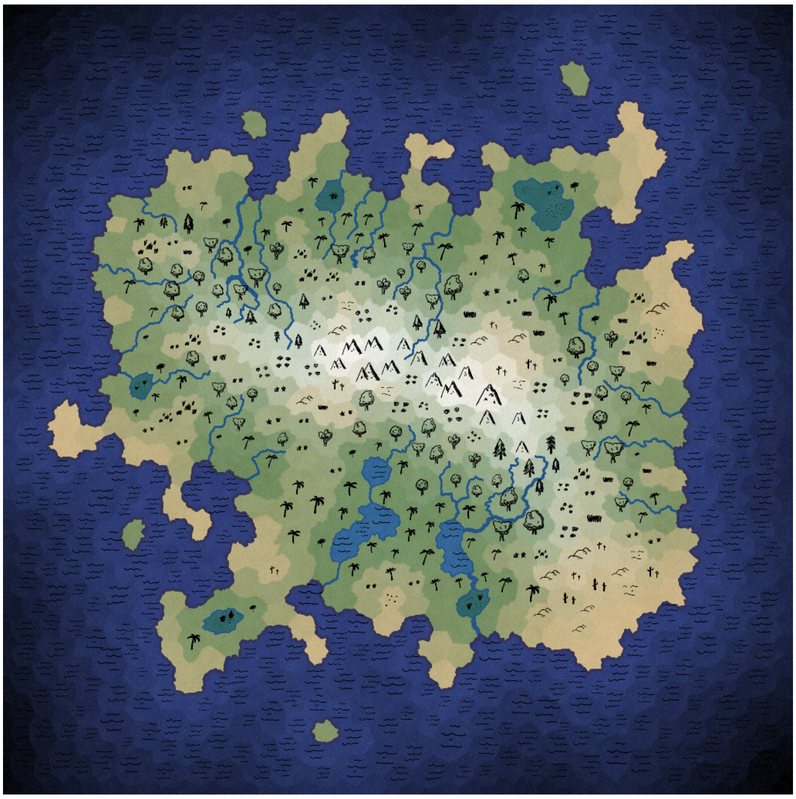
- KDevelop
- Code::Blocks
- PyCharm Professional
- WebStorm
- IntelliJ Ultimate
- Wing IDE 101
- Eclipse IDE for Enterprise Java Developers
  - GitHub Mylyn Connector
  - FindBugs
  - M2Eclipse
- Apache Maven (in /opt/)
- Android Studio
  - Android SDK API 29 (in /opt/Android/Sdk/)
  - KVM (equivalent to Intel HAXM)
- Jupyter Notebook
- Vim
- SublimeText 4
- Visual Studio Code
- emacs

Documentation

Language documentation is available. Man pages are also available in your workstation as usual, just use the command `man`.

PROBLEMS

Problem A: Rectangulon



They called it the "*Battle of the Red Sea*," for it was the bloodiest and most massive battle the isle of *Rectangulon* had ever seen.

Rectangulon, an island with many kingdoms, was peaceful for as long as the ancient ones can remember. However, power and greed changed all that. Each kingdom raised an army and was ready for battle, no matter the cost!

Nevertheless, there was still some dignity left in this darkest of times, as all kingdoms decided to abide by the ancient rules of battle:

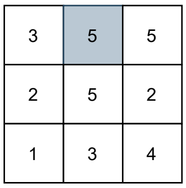
1. Whoever has the **largest** army on the island attacks **first**.
2. Always attack your **weakest** neighbor.
3. If two armies have the same size, **north** then **west** always takes precedence.
4. After a battle, the winning army must **all** move into the defeated army's position.

Which will be the last army standing?

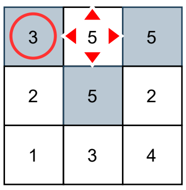
Task

Given the sizes of all armies in *Rectangulon*, calculate the place of origin and ending size of the last standing army of the "*Battle of the Red Sea*."

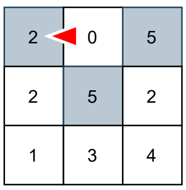
In each turn, the **largest** army is selected. If two armies are tied, the one that is further **north** is selected. If both are at the same latitude, the one further **west** is selected.



Then, the attacker chooses his **weakest** neighbor by looking into the four cardinal directions (north, west, east, and south). If two armies are tied, the one that is further **north** is selected. If both are at the same latitude, the one further **west** is selected.



The result of the battle is the difference between the sizes of both armies. If the attacking army has size s_1, and the defending army has size s_2, then the size of the attacking army will become $s_1 - s_2$, and the defending army will be destroyed (notice that $s_1 \geq s_2$ is always true). If both armies have the same size, both are destroyed. Otherwise, the remaining units of the attacking army move into the defender's army position.



If the selected army has no one to attack, then one of its units deserts due to boredom.

The last army standing wins the *Battle of the Red Sea*. Unless there are no survivors!

Input

The first line contains two integers, w and h, representing the width and height of the island.

The next h lines, each contain w integers representing the size, s, of each army.

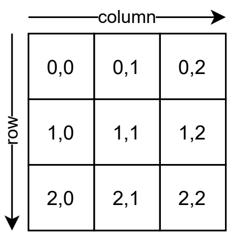
Constraints

$1 \leq w, h \leq 100$ Dimensions of the island
$1 \leq s \leq 1000$ Size of each army (number of units)

Output

A single line containing the string "none wins" if the last two armies destroy each other.

Otherwise, a single line containing three integers. The first two represent the original coordinates (row, column) of the last standing army, and the third represents the size of that army at the end of the battle. The origin of the map is the most northwestern kingdom with coordinates (0, 0).



Sample Input 1

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

Sample Output 1

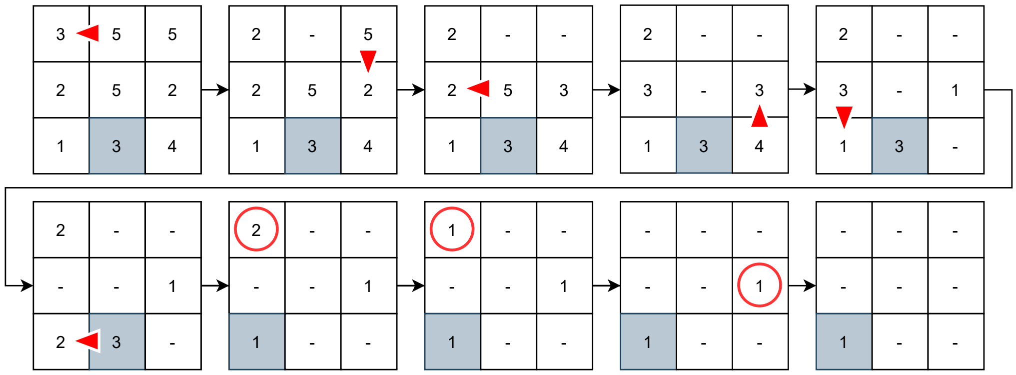
```
2 1 1
```

Sample Explanation 1

The following is a pictorial explanation of this sample input.

The grayed out square represents the kingdom that eventually wins the battle as it moves through the island.

Each 3×3 grid represents a step in the battle. The red arrow shows a battle between the largest army and its weaker neighbor. The red circle represent the largest army getting bored and losing a unit.



As you can see, the winning army ended at coordinates $(2, 0)$, but what we want are its starting coordinates, in this case $(2, 1)$.

Sample Input 2

```
3 3
5 10 5
5 5 5
10 5 10
```

Sample Output 2

```
none wins
```

Problem B: Where to Sit

The new school year is starting, and John is very excited to see his colleagues and friends again. However, he faces a dilemma. Whom is he going to sit next to?

John decided that he would sit at the same distance, using the Manhattan distance, from each of his best friends, and so he decided to create a little program that would allow him to determine where he should sit.

Task

Your task is to determine where John should sit. To achieve this, consider that the room has tables distributed on an $m \times n$ grid (m represents the rows and n the columns) and that tables in the room are at the same distance from each other, both vertically and horizontally, and that seats are numbered from left to right and from top to bottom starting at 1 (top left) and ending at $m \times n$ (bottom right), see Figure 1.

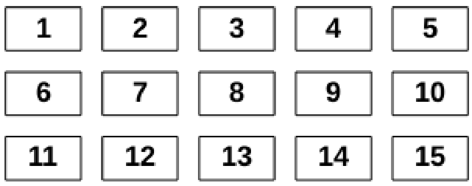


Figure 1: Example of a classroom distribution ($m = 3$ and $n = 5$)

Input

The input has the following format. In the first line, we have two integers, m rows and n columns, representing the grid dimension. In the second line, a single integer k represents the number of John's best friends. In each of the remaining k rows, a single integer s_i indicates the place where the i^{th} John's best friend sits. There are no repetitions.

Constraints

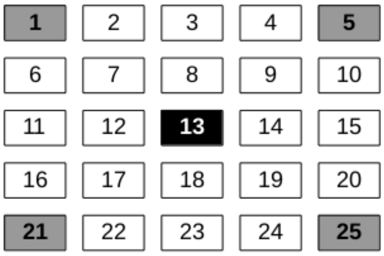
$1 \le m, n \le 100$	Number of rows and columns
$2 \le k \le min\{10, m \times n - 1\}$	Number of John's best friends
$1 \le s_i \le m \times n$ and $1 \le i \le k$	table where the i^{th} John's best friend sits

Output

The output consists of a positive number, between 1 and $m \times n$, representing the table where John should sit, or -1 if there is no possible solution. If there is more than one table at which John can sit, the one with the lowest number should be chosen.

Sample Input 1

```
5 5
4
25
5
21
1
```

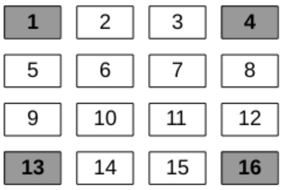


Sample Output 1

```
13
```

Sample Input 2

```
4 4
4
4
1
16
13
```

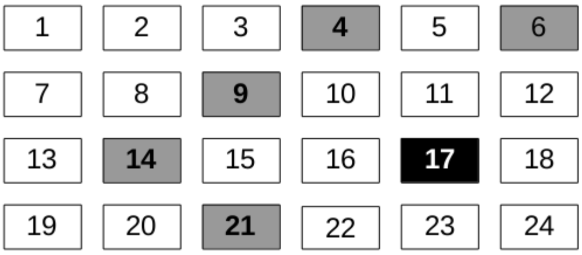


Sample Output 2

```
-1
```

Sample Input 3

```
4 6
5
4
9
21
14
6
```

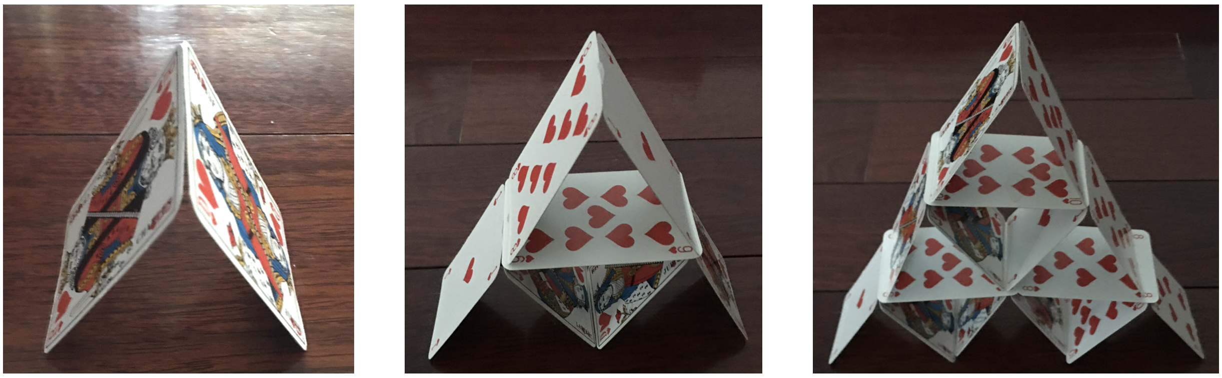


Sample Output 3

```
17
```

Problem C: House of Cards

Ahhhh, lockdown, too much spare time. Alice wants to build houses of cards. For a house with a single level, she only needs two cards, as depicted in Figure 1(a). For a house with two levels, she needs seven cards, and for a house with three levels, she needs fifteen cards, as we can see in Figure 1(b) and Figure 1(c), respectively.



(a) *One level* (b) *Two levels* (c) *Three levels*

Figure 1: Houses of cards with different levels.

Task

If we give Alice N cards, how many complete levels will her house of cards have?

Input

The input contains a single integer N representing the total number of cards given to Alice.

Constraints

$1 \leq N \leq 10^{15}$ Number of cards

Output

One integer corresponding to the highest number of levels of the built house with at most N cards.

Sample Input 1

```
15
```

Sample Output 1

```
3
```

Sample Input 2

```
17
```

Sample Output 2

```
3
```

Problem D: Pizza for Dinner

Alan and Ben are happy: There's pizza for dinner! They will play again! A few months ago, their parents devised some rules for sharing a pizza, putting an end to the unpleasant complaints of "His slice is always bigger than mine!".

Mum and Dad remove the first two slices for themselves and decide which son plays first. In each play, a child takes a slice of pizza and chooses who eats it.

- If the slice is for himself, his brother plays next.
- If the slice is given to the brother, it must be one of the largest available slices, but the (same) child is the next to play.

Of course slices have to be removed in a polite order: once Mum takes the first slice, opening a hole in the pizza, the available slices are those adjacent to the hole.

Ben, who is the youngest, almost never wins. Alan ends up eating more pizza. So, today he will try a new strategy: he will always take a largest slice, alternating between eating it and giving it to his brother. In his first play, he will eat the slice, in his second play, he will give the slice to Alan, in his third play, he will go back to eating the slice, and so on.

Task

Write a program that, given the sizes of the pizza slices, computes the maximum amount of pizza Ben can eat today (always taking a largest slice, and alternating between eating it and giving it to his brother), considering that Alan plays optimally (i.e. he always ends up eating the maximum possible amount of pizza). Notice that Mum can pick any slice, Dad can remove one of the two slices adjacent to that of Mum's, and the parents can choose Alan or Ben to play first.

Input

The input first line has a single integer, N, representing the number of pizza slices. The second line contains N integers, $S_1\, S_2\, \ldots\, S_N$, which denote the sizes of the slices. Notice that the pizza is circular, that is, slices 1 and N are adjacent to each other.

Constraints

$4 \leq N \leq 2500$	Number of pizza slices
$1 \leq S_i \leq 100$	Size of a pizza slice (for $i = 1, 2, \ldots, N$)

Output

The output consists of a single line with an integer, representing the maximum amount of pizza Ben can eat if he always takes a largest slice, alternating between eating it and giving it to his brother, and Alan plays optimally.

Sample Input

```
7
30 25 36 35 16 20 35
```

Sample Output

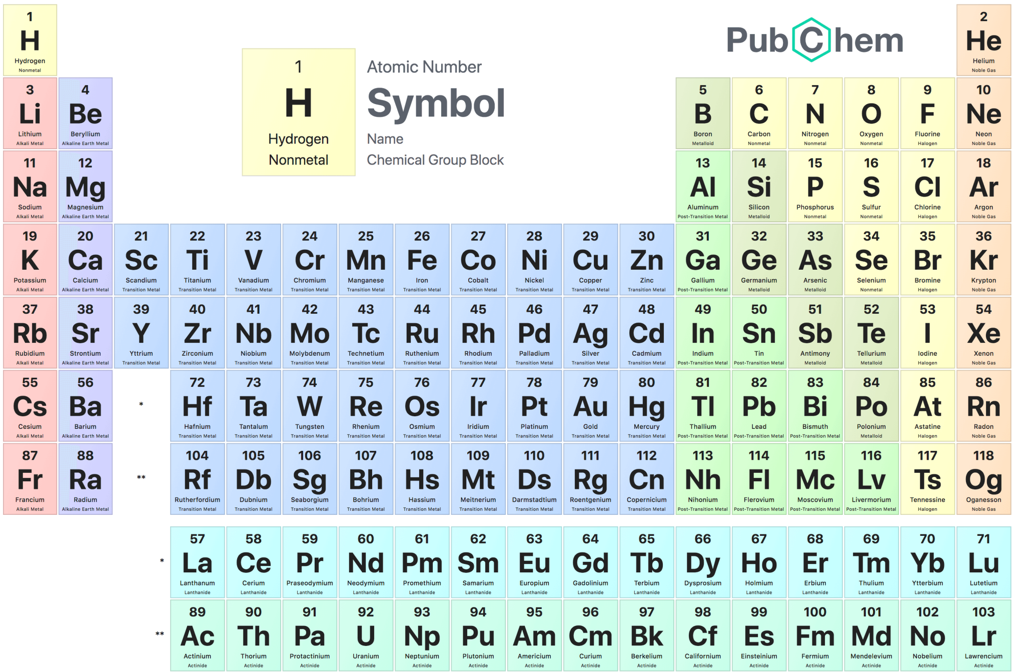
```
66
```

Sample Explanation

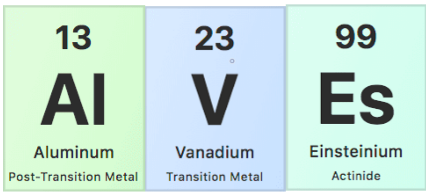
Ben eats the maximum amount of pizza when Mum removes the smallest slice, Dad picks the (adjacent) slice of size 35, Ben plays first, and the sequence of moves is the following:

1. Ben removes and eats the slice of size 36 (because this is his first play and $36 > 20$).
2. Alan (who plays optimally) decides to take the slice of size 20 for himself.
3. Then, Ben removes the slice of size 35 and gives it to Alan.
4. In his third play, Ben takes the larger slice (of size 30) for himself.
5. Alan takes (and eats) the only remaining slice (of size 25).

Problem E: Chemical Names



The other day I received an email from a colleague of the department of chemistry. In the signature section there was this nice little image:



It took me a moment or two to understand. The image is made up by the three little squares of the elements, aluminium, vanadium and einsteinium, taken from the periodic table. The symbols for those elements, `Al`, `V` and `Es`, make up `"Alves"`, which is the surname of my colleague.

How cool is that?!

I immediately tried to find the elements that compose my own name, `"Valente"`, with symbols of elements but, alas, it is not possible. I am not a chemist, but I am a programmer. I rushed to write a program that, given a name, computes the sequence of elements whose symbols make up that name, when that is possible.

Task

Please write a program that, given a sequence of names, displays for each of those names all the sequences of elements whose symbols make up the name.

Input

The first line of the input contains a positive integer, N, representing the number of names to be processed. N lines follow, each containing one name.

Constraints

$1 \leq N \leq 100$	Number of names
$1 \leq l_i \leq 15$	Length of each name
c	Characters used: each name in the input uses only lowercase Latin letters, without diacritics.

Output

For each name read, the program shall write one line echoing the name, followed by one line for each sequence of elements whose symbols compose that name. In each sequence the elements are represented by their names (not their symbols), each separated from the next by one space. The names shall be written in capitalised form, as they appear in the periodic table. Each group of lines, following the line that echoes the name that was read, shall be lexicographically sorted by the atomic weights.

Sample Input

```
6
alves
valente
costa
mcclane
biden
backus
```

Sample Output

```
alves
Aluminium Vanadium Einsteinium
valente
costa
Carbon Oxygen Sulfur Tantalum
Carbon Osmium Tantalum
Cobalt Sulfur Tantalum
mcclane
Moscovium Carbon Lanthanum Neon
biden
```

```
backus
Boron Actinium Potassium Uranium Sulfur
Barium Carbon Potassium Uranium Sulfur
```

Periodic Table

For your convenience, the periodic table is given below in text format (feel free to copy from the HTML version):

```
1 H Hydrogen
2 He Helium
3 Li Lithium
4 Be Beryllium
5 B Boron
6 C Carbon
7 N Nitrogen
8 O Oxygen
9 F Fluorine
10 Ne Neon
11 Na Sodium
12 Mg Magnesium
13 Al Aluminium
14 Si Silicon
15 P Phosphorus
16 S Sulfur
17 Cl Chlorine
18 Ar Argon
19 K Potassium
20 Ca Calcium
21 Sc Scandium
22 Ti Titanium
23 V Vanadium
24 Cr Chromium
25 Mn Manganese
26 Fe Iron
27 Co Cobalt
28 Ni Nickel
29 Cu Copper
30 Zn Zinc
31 Ga Gallium
32 Ge Germanium
33 As Arsenic
34 Se Selenium
```

```
35 Br Bromine
36 Kr Krypton
37 Rb Rubidium
38 Sr Strontium
39 Y Yttrium
40 Zr Zirconium
41 Nb Niobium
42 Mo Molybdenum
43 Tc Technetium
44 Ru Ruthenium
45 Rh Rhodium
46 Pd Palladium
47 Ag Silver
48 Cd Cadmium
49 In Indium
50 Sn Tin
51 Sb Antimony
52 Te Tellurium
53 I Iodine
54 Xe Xenon
55 Cs Caesium
56 Ba Barium
57 La Lanthanum
58 Ce Cerium
59 Pr Praseodymium
60 Nd Neodymium
61 Pm Promethium
62 Sm Samarium
63 Eu Europium
64 Gd Gadolinium
65 Tb Terbium
66 Dy Dysprosium
67 Ho Holmium
68 Er Erbium
69 Tm Thulium
70 Yb Ytterbium
71 Lu Lutetium
72 Hf Hafnium
73 Ta Tantalum
74 W Tungsten
75 Re Rhenium
76 Os Osmium
```

```
77 Ir Iridium
78 Pt Platinum
79 Au Gold
80 Hg Mercury
81 Tl Thallium
82 Pb Lead
83 Bi Bismuth
84 Po Polonium
85 At Astatine
86 Rn Radon
87 Fr Francium
88 Ra Radium
89 Ac Actinium
90 Th Thorium
91 Pa Protactinium
92 U Uranium
93 Np Neptunium
94 Pu Plutonium
95 Am Americium
96 Cm Curium
97 Bk Berkelium
98 Cf Californium
99 Es Einsteinium
100 Fm Fermium
101 Md Mendelevium
102 No Nobelium
103 Lr Lawrencium
104 Rf Rutherfordium
105 Db Dubnium
106 Sg Seaborgium
107 Bh Bohrium
108 Hs Hassium
109 Mt Meitnerium
110 Ds Darmstadtium
111 Rg Roentgenium
112 Cn Copernicium
113 Nh Nihonium
114 Fl Flerovium
115 Mc Moscovium
116 Lv Livermorium
117 Ts Tennessine
118 Og Oganesson
```

Problem F: Reverse the Numbers

4329581 - 1859234 = ??

Chris likes puzzles and is asking for your help with this problem: given a set of digits, how should he arrange the digits such that the absolute difference between the number and the same digits in reverse order is minimized? For example, with the digits 123 in this order, the absolute difference between the reverse and the original number is $321 - 123 = 198$. But this is not the best answer, as Chris could use the same three digits to form the number 132 and get a difference of $231 - 132 = 99$, which is smaller.

Task

Your task is to compute the smallest possible absolute difference between a number and the reverse, for a given set of digits.

Input

The input starts with the number n of digit sequences to evaluate. Then, each of the following n lines has a number in base 10, determining the base b, followed by a sequence of digits in this base b that you should arrange in any order to form the number (0 is allowed in the beginning). Each digit can show up at most once and each sequence has at least 2 digits. To represent digits up to base 36, we consider the following table:

Digit	Value in base 10
0	0
1	1
...	...
9	9
A	10
B	11
C	12
...	...
V	31
W	32
X	33
Y	34
Z	35

Constraints

$1 \leq n \leq 10$ Number of sequences to evaluate

$2 \leq b \leq 36$ Base of the sequence

Output

For each of the n digit sequences, you should output the smallest absolute difference, between a number using all the provided digits and its reverse. You should output one difference per line in base 10.

Sample Input 1

```
1
10 123
```

Sample Output 1

```
99
```

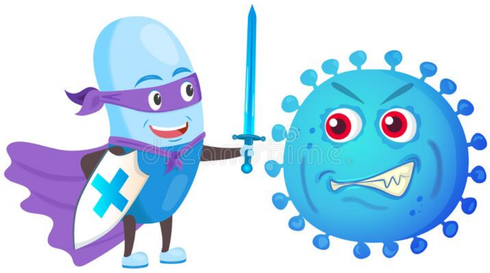
Sample Input 2

```
4
20 91EG30DHBF7I2C8A5
9 063145
27 4ACENHFL6J1BIK2Q739O0MDP
30 690EQN125PO4H
```

Sample Output 2

```
38123661317534838759
17144
246859741217362690444220298310 4
56241373350406379
```

Problem G: World War Viruses



The battle of our age is upon us. Viruses against anti-bodies! The two armies will meet in an epic encounter to determine the faith of the world as we know it. Each army is composed of fearless soldiers who will defend their cause at any cost. The micro-organisms follow the Roman discipline and fight using very accurate tactics. What will be the outcome of this historical confrontation?

Task

Your task is to determine whether there will be an army which emerges victorious, the fight goes on forever, or the strife ends up in a draw. The rules are simple: initially both armies have the same number of elements, each one disposing of a fighting power represented by an integer value. The fight is very disciplined. The soldiers form a line and only the first elements of each army fight between them. One of three things can happen:

1. The anti-body has a higher fighting power than its opponent. In such a case, the virus is converted into an anti-body conserving its original fighting power. At the end of the round, both the victorious anti-body and the newly converted anti-body return to the tail of their army, in this order.

2. The virus has a higher fighting power than its opponent, hence everything happens inversely when compared with the previous case. The only exception is the order in which the soldiers return to the tail of the army: first, the newly converted virus and only then the original virus.

3. The two fighters have the same fighting power. In such a case, the armies will use the *tower* strategy: the two original fighters stay in the base of the structure, a new virus and a new anti-body climb into the shoulders of their respective friend and finally a third soldier from each army climbs into the top of the tower. At this point, we compare the fighting power of those on top. If the anti-body wins, step (1) applies and both towers are appended to the anti-body army; if the virus wins, step (2) applies

(again, both towers are appended to the victorious army); finally, if both have the same power, we repeat step (3). The tower-formation process ends as soon as one of the armies runs out of soldiers, in which case the other army wins the war, or if they both run out of soldiers, in which case a draw is declared.

Input

The first line contains N, the number of elements in each army, followed by $2N$ lines. The second line of the input represents the fighting power of the first anti-body, the third line of the input represents the fighting power of the second anti-body, and so on. Line $N+2$ represents the fighting power of the first virus, line $N+3$ represents the fighting power of the second anti-body, and so on. The last N lines contain the fighting power of viruses.

Constraints

$0 \leq N \leq 1000$	Number of soldiers in each army
$0 \leq AB_i \leq 1000$	Fighting power of each anti-body
$0 \leq V_i \leq 1000$	Fighting power of each virus

Output

A single line containing the result of the battle:

- `ANTI-BODY` means the anti-bodies win;
- `VIRUS` means the viruses have won;
- `DRAW` means the battle ends as a draw;
- `INFINITE` means the battle goes on forever;

Sample Input 1

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

Sample Output 1

```
ANTI-BODY
```

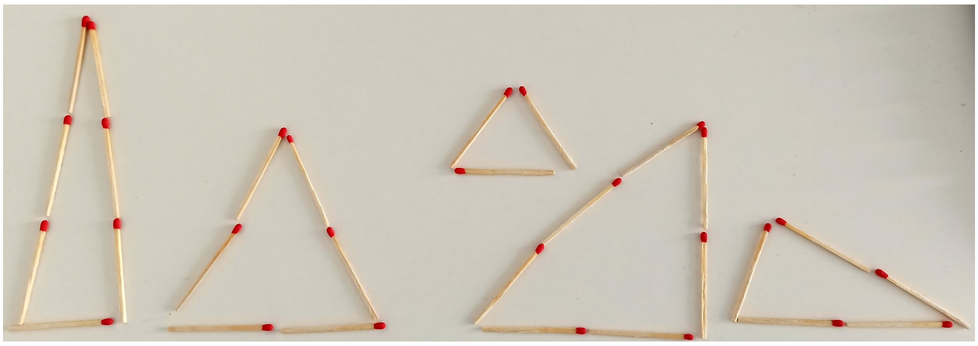
Sample Input 2

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

Sample Output 2

```
DRAW
```

Problem H: Number of Triangles



Charles and Diane love to defy themselves with mathematical challenges. This time, Diane gave the following challenge to Charles: *If I give you a box with M matches of the same size, how many different triangles can you make using them? You don't need to use all the matches.* Charles couldn't find a general solution for this problem and asked for your help. Can you create a program that is capable of solving this problem?

Each side of the triangle can use one or more matches. For instance, given 7 matches, you can create a triangle with two sides using 3 matches and the remaining one using 1 match or one side with 3 matches and the other two with 2 matches. Of course there are more solutions if you don't use all the matches:

- 3 3 1
- 3 2 2
- 2 2 2
- 2 2 1
- 1 1 1

Thus, there are 5 different ways of using up to 7 matches for creating triangles.

Task

Your task is to write a program that is capable of answering this question.

Input

Your program accepts the number of matches M.

Constraints

$3 \leq M \leq 10000$ Number of matches

Output

Your program should print the number of valid triangles that can be made with up to M matches.

Sample Input 1

```
3
```

Sample Output 1

```
1
```

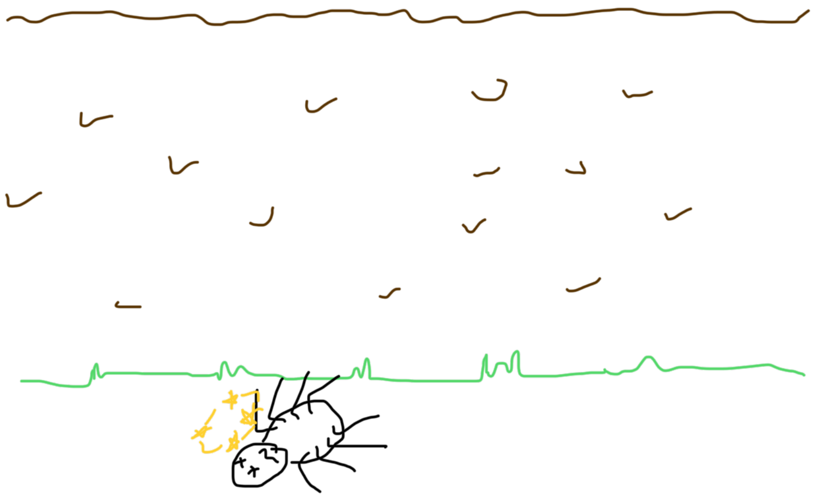
Sample Input 2

```
19
```

Sample Output 2

```
64
```

Problem I: Hill, the Climber

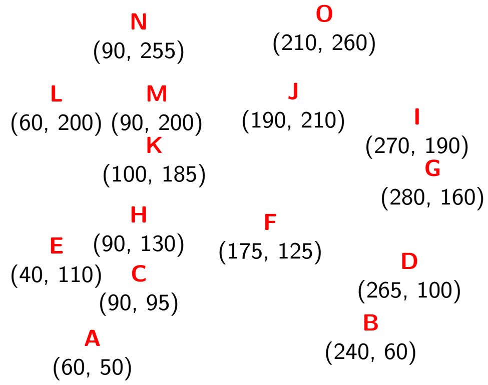


Hill, the Climber, likes to climb up everything. From chairs to buildings, from lamp posts to cliff walls, you name it, Hill climbs it. But Hill is a thoughtful and careful climber. Before starting to climb anything, Hill maps it and chooses the course to take up to the top. There is nothing worse than discovering that there is no route to the top when you are halfway up a mountain.

To plan a course for a climb, Hill precisely identifies and maps all the points which may be used as holds, during the ascent, sometimes with the aid of a drone. Afterwards, taking into account the distance between holds, Hill chooses a route to the summit. The best routes are those that pass through the least possible hold points.

The figure on the right depicts one such map (not completely to scale), of a 3.5 m high wall, where 14 hold points have been identified, whose coordinates are shown in centimetres.

The routes Hill may follow when climbing depend on Hill's *reach*, which is 90 cm. This means that Hill may only move from one hold point to another if they are 90 cm or less apart. For the same reason, the first hold point used must not be higher than 90 cm, and the last one must be located 90 cm or less from the top of the climb.



Given these constraints, Hill must start the climb up this wall at either hold point A or hold point B, and the final hold point must be O, which is the only one from which Hill is able to reach the top of the wall. In this case, Hill's sole best route starts at hold point A, and then passes through hold points H, F, J and O, before finally reaching the top, a total of 5 hold points.

Task

Your task is to help Hill compute the number of hold points on a best route for a climb to the top, given the height of the climb, the coordinates of all the hold points, and Hill's reach.

Input

The first line contains three integers, N, H and C, representing, respectively, the number of hold points, the height of the climb, in centimetres, and the number of test cases.

The following N lines contain a pair of integers standing for the (x, y) coordinates of the hold points, also in centimetres. The first coordinate of a hold point corresponds to its distance to some arbitrary reference line to the left of all hold points, and the second to its height, with respect to the base of the climb.

The final C lines contain one integer each, which represents the reach R of the climber, for each test case, in centimetres. You may assume that no more than around one hundred hold points are reachable from any hold point.

Constraints

$1 \leq N \leq 30\,000$	Number of hold points
$2 \leq H \leq 40\,000$	Height of the climb
$(0, 1) \leq (x, y) \leq (10\,000, H - 1)$	Coordinates of the hold points
$1 \leq C \leq 20$	Number of test cases
$1 \leq R \leq 200$	Climber's reach

Output

The output consists of one line for each of the test cases, containing either a single integer, denoting the number of hold points on a best route for the climb, or the word `unreachable`, if the given reach does not allow the climber to reach the top of the climb.

Sample Input

```
15 350 2
60 50
240 60
90 95
265 100
40 110
175 125
280 160
90 130
270 190
```

```
190 210
100 185
60 200
90 200
90 255
210 260
90
89
```

Sample Output

```
5
unreachable
```

Problem J: Palm Island Neighbours

The Palm Jumeirah is a man-made archipelago in Dubai. It is located on the coast of Dubai, United Arab Emirates. The shape of the island makes it so that two inhabitants close in geographical space still need to traverse a large path to visit each other. Unless they travel by boat, which is not allowed in this problem. We shall only consider paths that can be traversed on foot or by car. Preferably by car, because Dubai is usually extremely hot and it is unpleasant to go anywhere without air conditioning.

The exact distances are also not very relevant. We are only interested in how many neighbours we need to pass by until we reach a specific one. Note that the street connecting two neighbours always works both ways. Thus good neighbours make a huge difference in the quality of life.

Task

To determine how exotic the topology of this island is our goal is to compute what is the largest smallest path between two inhabitants. The input describes the topology of the island.

Input

The first input line contains the number n of inhabitants. The inhabitants are numbered from 1 to n. Each of the remaining $n-1$ lines has two different inhabitants, which indicates that there is a connection between them. Note that the island topology can not contain cycles, since it is a palm tree. Also, there is always a path between any two inhabitants, since each input considers only an island not the archipelago.

Constraints

$1 \leq n \leq 50\,000$ Number of inhabitants

Output

The output has a single integer, representing the number of connections on the shortest path between two inhabitants that are furthest away.

Sample Input 1

```
12
4 10
10 5
5 1
9 3
1 3
3 7
7 12
7 6
6 2
6 8
8 11
```

Sample Output 1

```
8
```

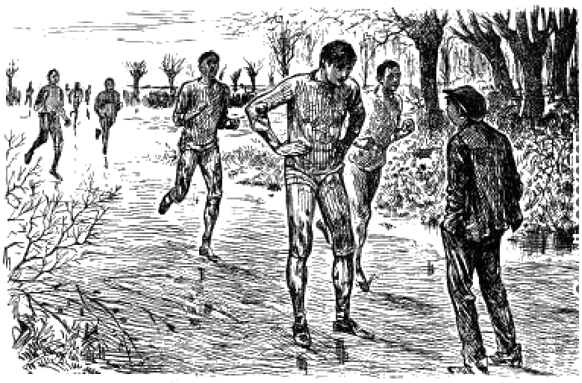
Sample Input 2

```
1
```

Sample Output 2

```
0
```

Problem K: The Rotten Paperchase Industry



HARE AND HOUNDS—AND DONKEY

" Seen two men with bags of paper pass this way ? "—" No ! " " Did they tell you to say no ? "—" Yes."

In the time we rediscover the outside world, adherence to outdoor social games exploded. Socializing in large spaces is the new normal. A good paper chase is so *post-confinement*!

In a post-confinement paper chase, contestants take a challenge notebook with all the necessary information for the possible routes and challenges. For each successful challenge, contestants are given a colorful ticket. At the finish line, contestants present the colored tickets in the order in which they won them. If this sequence corresponds to a valid configuration (a sequence of colors that ensures that the contestants have completed a complete victorious course), the contestants have successfully completed the race and the sequence is called a *winning sequence*.

A paper chase game is represented as a directed graph with N nodes and with labeled edges. The depart line is node 0 and the finish line is node $N-1$. Each edge is a geographically located challenge. A label c from, say, a node A to a node B is the ticket to win when the contestants try to go from A to B. If they are able to solve the challenge they get the ticket c, reach node B and can now proceed from B.

Unfortunately, competition between daytime entertainment companies is fierce and some of them try to lower organization costs by proposing recycled or unoriginal routes.

Examples are routes that validate the same configurations as in previous games, saving costs in the checking process in the finish line. Thus, a malicious competitor, taking advantage of this knowledge, can simply, at no risk, present the tickets earned in previous editions and unfairly win the paper-chase.

Two paper-chase games are equivalent (and share the same checking process) if any winning sequence of tickets in one paper-chase is also a winning sequence in the other, and vice-versa.

Task

When asked to help the organization, you were assigned the task to write a program that decides whether two given games are equivalent.

Input

The input is constituted by the two graphs. The nodes of the graphs are integers, the tickets are lowercase letters. For each graph, the starting vertex is 0 and the finish line vertex is the vertex with the highest (integer) identifier.

The first graph is introduced as follow:

- The first line introduces the number N_1 of nodes of the graph. The nodes of this graph are then numbered from 0 to $N_1 - 1$.
- The second line introduces the number E_1 of edges of the graph.
- The following E_1 lines introduce an edge each, using the following format $n\ c\ m$, where n and m are the nodes ($0 \leq n, m < N_1$) and c is the ticket to win, represented as a lowercase letter (from 'a' to 'z').

The second graph is then introduced in the same way (N_2, E_2 and the E_2 lines).

Some details:

- given two nodes n_1 and n_2, one can have zero, one or more than one challenge. If there is more than one challenge (several edges between n_1 and n_2), the tickets are all distinct;
- two edges (say, from n_1 to n_2 and from v_1 to v_2) can share the same ticket (the same label) as soon as $n_1 \neq v_1$ or $n_2 \neq v_2$.
- a game always has a departing point and an arrival point and every path leads and ends to the arrival node.

Constraints

$2 \leq N_i \leq 150$	Number of nodes in graph $i \in \{1, 2\}$
$0 \leq E_i \leq 500$	Number of edges in graph $i \in \{1, 2\}$
c	Each ticket is represented by a single ASCII lowercase letter $\in [a - z]$

Output

A single line with the word `PROBLEM`, if the two games are equivalent, or with the sentence `NO PROBLEM`, otherwise.

Example 1

Consider the following two games (Figures 1 and 2), involving 8 nodes.

The two games are in fact equivalent. Each winning sequence of tickets in one game is also a winning sequence in the other.

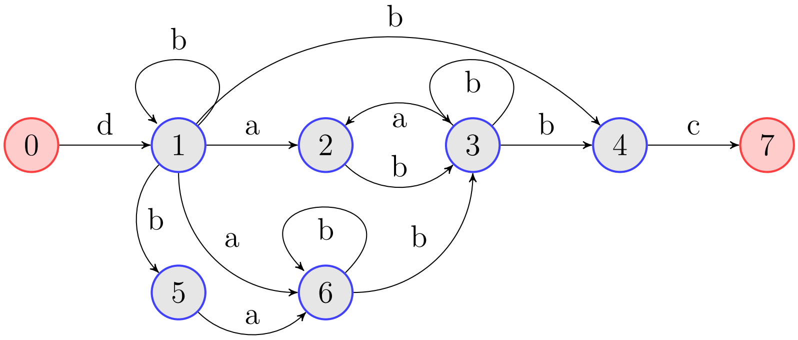


Figure 1: First game (Example 1)

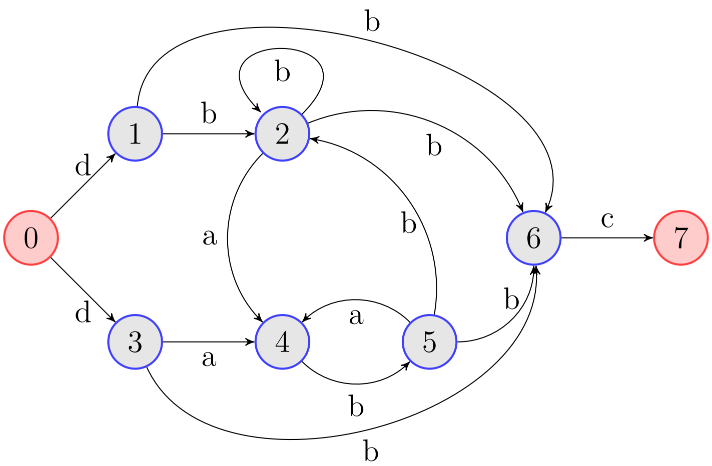


Figure 2: Second game (first example)

Example 2

On the contrary, the following two games (Figures 3 and 4) are not the same.

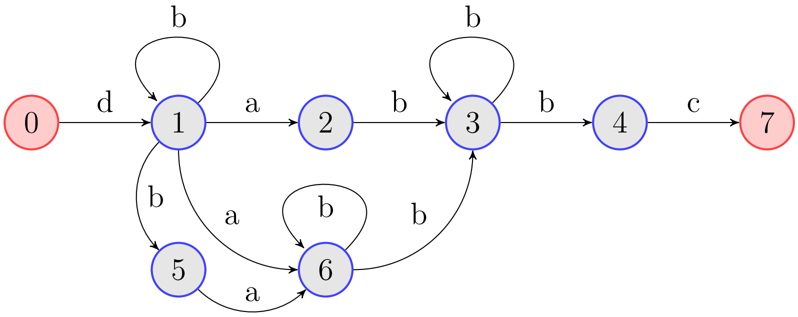


Figure 3: First game (second example)

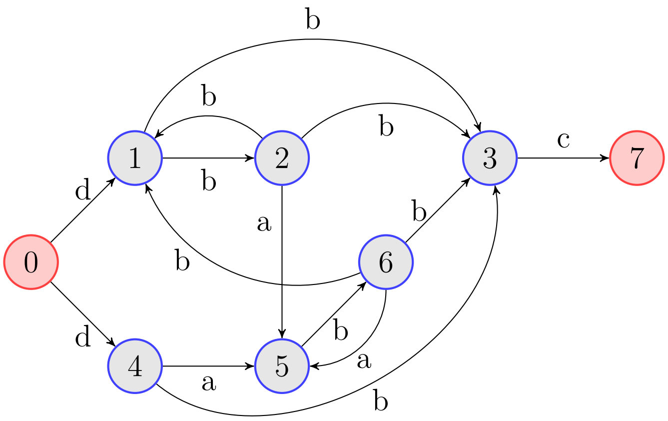


Figure 4: Second game (Example 2)

Sample Input

```
8
14
0 d 1
1 b 1
1 a 2
1 b 5
1 a 6
2 b 3
```

```
3 b 3
3 b 4
4 c 7
5 a 6
6 b 3
6 b 6
8
14
0 d 1
0 d 4
1 b 2
1 b 3
2 a 5
2 b 1
2 b 3
3 c 7
4 a 5
4 b 3
5 b 6
6 a 5
6 b 1
6 b 3
```

Sample Output

```
NO PROBLEM
```