培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE

第三屆 · 2018

2018 年 5 月 22 日 (星期二)

香港培正中學

題號		名稱	時間限制	記憶體限制
A		Ambidexterity	1 秒	256 MB
B		Bob the Builder	2 秒	256 MB
C		Consecutive Numbers	1 秒	256 MB
D		Differentiation	1 秒	256 MB
E		Error	1 秒	256 MB
F		Final Fixture	2 秒	256 MB
G		Go	1 秒	256 MB
H		Handicap	1 秒	256 MB
I		Infection	1 秒	256 MB
J		Jakanda Forever	2 秒	256 MB
K		Kids' Entertainment	1 秒	256 MB
L		Labyrinth	1 秒	256 MB

Pascal	`ppcx64-3.0.0 -O2 -Sg -v0 -dEVAL -XS`
C	`gcc-5.3.1 -static -Wno-unused-result -DEVAL -lm -s -O2`
C++	`g++-5.3.1 -static -std=c++11 -Wno-unused-result -DEVAL -lm -s -O2`

編繹時間限制 10 秒　編繹記憶體限制 512 MB

作者　周君珽 董鑫泉 黃敏恆 黃梓駿 黃亦駿

NOTES

1. Output strings are checked case-sensitively.
2. Trailing spaces in output lines are allowed.
3. The end-of-line character at the end of the output is optional.
4. Using scientific notation for real number output is not allowed.
5. 64-bit integer types (e.g. `int64` for Pascal, `long long` for C/C++) may be required for some problems. The C format string token is `%lld`.

A Ambidexterity

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
第三屆 · 2018年5月22日

In this task, you are going to determine whether Nai is *ambidextrous*, i.e. whether he can write equally well with both hands.

You have learned that Nai writes at speed L characters per minute with his left hand and at speed R characters per minute with his right hand. If $L = R$, then obviously Nai writes equally well with both hands; but we allow some leniency.

Given a parameter P, where $50 \leq P \leq 100$, Nai is considered ambidextrous if the writing speed of his weaker (slower) hand is at least $P\%$ that of his dominant (faster) hand, or if $L = R$, in which case it is hard to tell which hand is dominant. Otherwise, he is either left- or right-handed.

This simple task should not take you more than a few minutes, right?

Input

The first and only line of input consists of three space-separated integers L, R, and P.

For all test cases, $50 \leq P \leq 100$, $1 \leq L, R \leq 100$.

Output

If Nai is ambidextrous, output `Ambidextrous`.

Otherwise, if Nai's dominant hand is his left hand, output `Left-handed`. Otherwise, output `Right-handed`.

Examples

input	output
18 37 50	Right-handed

input	output
19 37 50	Ambidextrous

input	output
100 99 100	Left-handed

This page is intentionally left blank.

B Bob the Builder

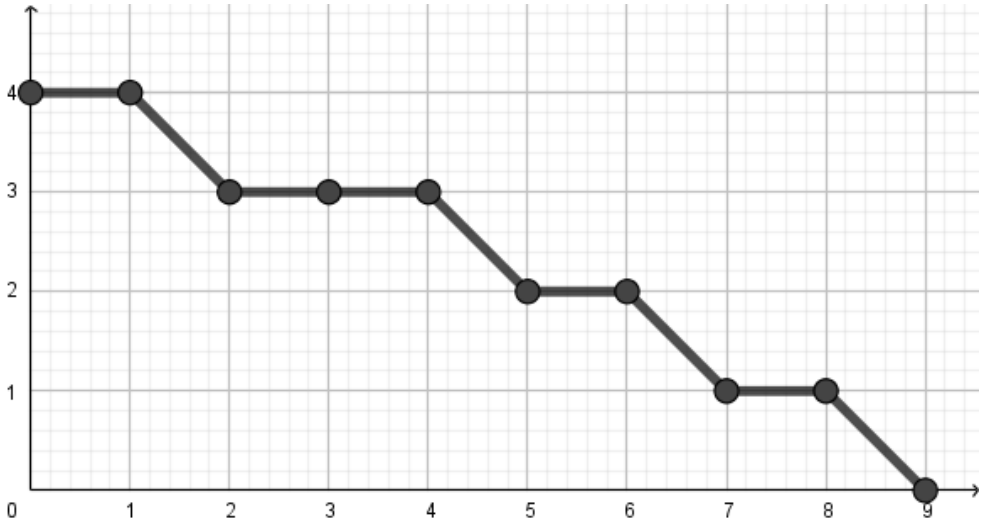
Time Limit: 2 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Bob the builder wants to build the most exciting roller coaster in the world!

Due to the limited space, Bob can only build the railroad without turning left or right. Also, Bob thinks the only interesting part of roller coaster is going down, so he will not consider any parts of the railroad going upward as well.

Under the above conditions, you may consider the railroad as a polyline (broken line) on a 2D Cartesian coordinate plane. Formally, Bob has to build the railroad from $x = 0$ to $x = N$. For each integer $x = i$ ($0 \le i \le N$), Bob can choose the height H_i of the railroad as any **non-negative integer**. In other words, the railroad is a polyline formed by the points $(0, H_0), (1, H_1), \ldots, (N, H_N)$ (in this order), with the constraints that:

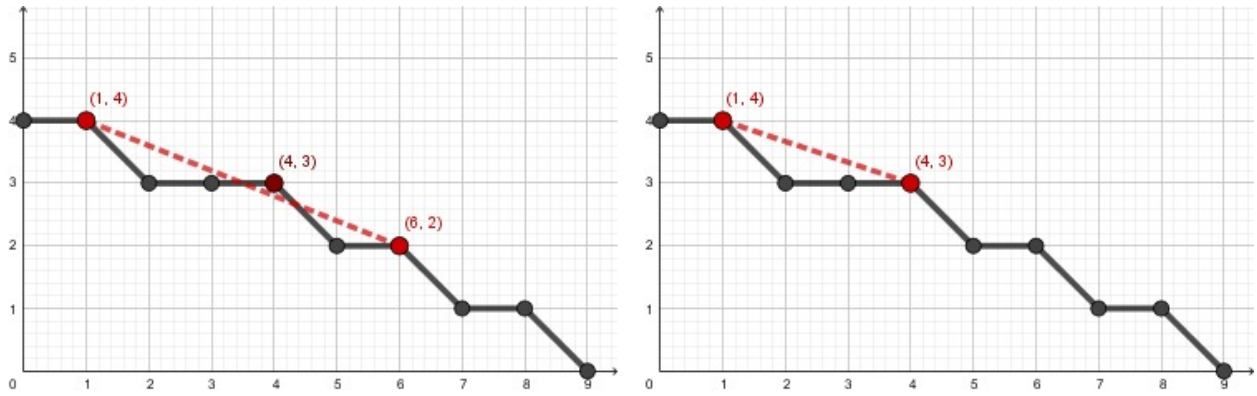
- For safety reasons, the railroad should not be too steep, so for integers i ($0 \le i < N$), $H_{i+1} = H_i$ or $H_{i+1} = H_i - 1$.
- To let the passengers leave on the ground, $H_N = 0$.



An example with $N = 9$ and $H_{0..9} = \{4, 4, 3, 3, 3, 2, 2, 1, 1, 0\}$

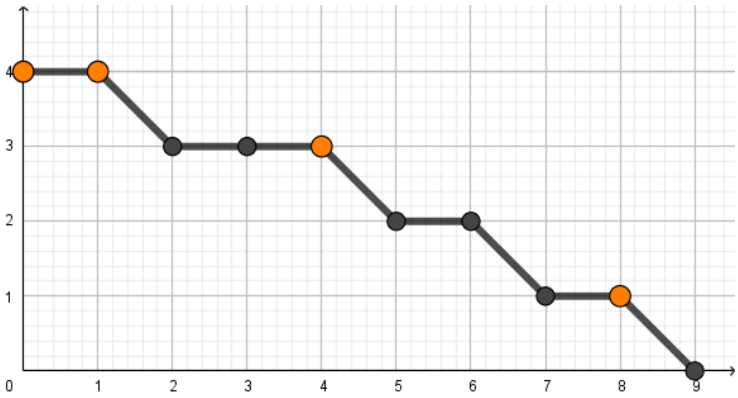
To build an exciting roller coaster, Bob defines the *exciting index* as the number of times the passengers seeing new furthest part of the railroad. Formally:

- The passengers ride in positive x direction (i.e. from $x = 0$ to $x = N$), along the railroad.
- For integers i ($0 \le i \le N$), when passengers are at the point (i, H_i), the furthest part of railroad they can see is defined the largest integer f_i ($i \le f_i \le N$) such that there **DO NOT EXIST** any integers j ($i < j < f_i$) that the point (j, H_j) is **strictly above** the line segment formed by (i, H_i) and (f_i, H_{f_i}).



In this example, passengers at the point $(1, H_1)$ cannot see the point $(6, H_6)$ as there exists point $(4, H_4)$ lying strictly above the line segment formed by $(1, H_1)$ and $(6, H_6)$. However, they can see the point $(4, H_4)$ as there are no points lying strictly above the line segment formed by $(1, H_1)$ and $(4, H_4)$.

- For integers i ($0 \leq i \leq N$), if the value of f_i is **strictly greater** than f_k for all integers k ($0 \leq k < i$), then point (i, H_i) is considered as an *exciting point*. In particular, $(0, H_0)$ is an *exciting point*.
- The *exciting index* of the railroad is the number of *exciting points*.



An example with $f_{0..9} = \{1, 4, 4, 4, 8, 6, 8, 8, 9, 9\}$ and *exciting index* = 4

(Note that $f_4 = 8$ as $(6, 2)$ is just lying on the line segment formed by $(4, 3)$ and $(8, 1)$, but NOT "strictly above")

Given the value of N, please write a program to help Bob finding any valid design maximizing the *exciting index*.

Input

The only line contains a single integer N ($1 \leq N \leq 10^6$).

Output

On the first line, output the maximum *exciting index*.

On the next line, output $H_0, H_1, \ldots H_N$, representing the design of the railroad.

Example

input	output
9	4 4 4 3 3 3 2 2 1 1 0

C Consecutive Numbers

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

"Captain Teemo on duty!"

"Hut two three four" Perhaps Teemo loves consecutive numbers as many people do.

Tristana, as a lover of consecutive numbers (and also Teemo), asked you Q questions. For each question, you have to determine whether the given number X can be represented as a sum of at least two consecutive positive integers. For example, the answer for $X = 9$ is `Yes` as $9 = 2 + 3 + 4$ while the answer for 2 is `No` (note that $(-1) + 0 + 1 + 2 = 2$ is invalid as only positive integers are allowed).

Input

The first line contains a single integer, Q, denoting the number of questions Timo asked ($1 \leq Q \leq 10^5$).

In each of the next Q lines, a positive integer X is given ($1 \leq X \leq 10^9$).

Output

For each of the Q questions, print the answer in a single line: either `Yes` or `No`.

Example

input

```
6
1
2
10
20
100
200
```

output

```
No
No
Yes
Yes
Yes
Yes
```

This page is intentionally left blank.

D Differentiation

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

Write a program to differentiate a polynomial in x.

Here's how to differentiate a polynomial in the form

$c_n x^n + c_{n-1} x^{n-1} + \cdots + c_1 x + c_0$

Each term $c_i x^i$ where $i \geq 1$ can be handled individually:

1. Multiply the coefficient c_i by i.
2. Decrease the exponent of x by 1.
3. Therefore, the result of the term is $(c_i \times i) x^{i-1}$

Finally, join the results of the terms together to form the final polynomial. Also, the derivative of the constant term is 0.

For example, the derivative of $-3x^3 + 5x^2 - 7x + 4$ is $(-3 \times 3)x^{3-1} + (5 \times 2)x^{2-1} + (7 \times 1)x^{1-1} + 0$ = $-9x^2 + 10x - 7$

The polynomial's format is similar to the requirements for math homework:

- The polynomial does not start with a plus sign.
- The polynomial contains no spaces.
- The terms should be ordered in strictly decreasing degree of x.
- There should be no redundant terms. The term is omitted if the coefficient is 0.
- The `1` is omitted if the coefficient is 1 or -1. (see sample 1)
- The degree is written directly after `x`. The exponent is omitted if the degree is 1.
- The constant term is omitted if it is zero, except that when the whole polynomial is zero (see sample 3).

Therefore, there is only one way to format a polynomial correctly.

Input

The input consists of a string: the polynomial. The maximum degree of x is 99 and the coefficients are integers between -99 and 99, including -99 and 99 but excluding 0. The length of the string is at most 100.

It's guaranteed that the string strictly follows the format, i.e. `+0x1-1x2+0` is not a possible input.

Output

Output the derivative of the polynomial. It must strictly follow the format.

Examples

input	output
4x5-x2	20x4-2x
-6x+3	-6
5	0
-3x3+5x2-7x+4	-9x2+10x-7

E Error

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

Percy is a primary school student. He is learning how to do addition and subtraction. Being a smart boy, he decided to use a calculator instead of doing it by hand.

The calculator can only store and display non-negative integers of up to 8 digits, i.e. integers between 0 and 99999999 inclusive. If at any time the result of an operation exceeds this range, the calculator will display `Error` and stops functioning. Therefore, Percy tries to rearrange the terms in the expression he is trying to calculate.

Specifically, the expression contains N numbers between -99999999 and 99999999 and Percy needs to sum them up using the calculator one by one. Since numbers can be summed up in any order, help Percy find an order that won't cause any error.

Input

The first line contains an integer N, the number of integers that Percy needs to sum up ($1 \leq N \leq 10$).

Each of the next N lines contains an integer between -99999999 and 99999999 inclusive.

Output

If it is not possible to calculate the sum of the integers without causing calculator error, output `Error`.

Otherwise, in separate lines output ANY order of the N integers that would not cause calculator error. The integers shall be added to / subtracted from the calculator one by one from top to bottom.

Examples

input

```
4
33334444
87654321
-10000000
-20000000
```

output

```
33334444
-10000000
-20000000
87654321
```

input

```
3
-100
-200
-300
```

output

```
Error
```

input

```
3
-400
10000
-7000
```

output

```
10000
-7000
-400
```

Note

In sample 1, one of the possible order is +33334444 → -10000000 → -20000000 → +87654321. The results after the operations are 33334444, 23334444, 3334444 and 90988765. All of them are between 0 and 99999999 inclusive. Note that +87654321 → -20000000 → -10000000 → +33334444 is also acceptable. However, +87654321 → -20000000 → +33334444 → -10000000 is not an acceptable answer because the result after +33334444 is 100988765 which exceeds 99999999.

In sample 2, it will cause Error no matter how Percy reorders the operations.

F Final Fixture

Time Limit: 2 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Like in many countries, there is a football league in Hackerland. The top league (called Tourist League) has N teams, where N is an even number. The teams are conveniently numbered from 1 to N.

For those unfamiliar with football leagues, here is the basic information that you need to know for this task. Three statistics determine a team's standing: points, goals scored, and goals conceded. For every match a team plays, they score goals and concede goals.

- If they score more goals than they concede, then they win and earn three points.
- If they score the same number of goals as they concede, then they draw and earn one point.
- If they score fewer goals then they concede, then they lose and earn no points.

For two teams A and B, let P_A be the total points team A has, S_A be the total number of goals scored by team A, and C_A be the total number of goals conceded by team A. Likewise for P_B, S_B, and C_B. Team A is ranked higher than team B if and only if one of the following is true:

- $P_A > P_B$ (more points)
- $P_A = P_B$ and $S_A - C_A > S_B - C_B$ (same points, superior goal difference)
- $P_A = P_B$, $S_A - C_A = S_B - C_B$, and $S_A > S_B$ (same points, same goal difference, more goals scored)

There is no further tiebreaker. Therefore, it is possible for two teams to have the same rank. A team's rank is given by $1+$ (number of teams ranked higher than it).

The final fixture in the league is always the most exciting one, especially at the top and at the bottom of the table, where teams fight fiercely for their respective targets. More important, all matches will start at the same time, and a team's fate may depend on scorelines of matches played hundreds of miles away!

For all N teams in the Tourist League, you know their points, goals scored, and goals conceded, right before the final fixture. You also know the pairing for the final fixture.

Here is your task: based on your data, determine each team's best and worst final rank. For the sake of this task, there is no upper limit on the number of goals a team can score within a match; even a $10^9 - 0$ scoreline is considered possible.

The given data (points, goals scored, goals conceded) may be inconsistent, since they come from a questionable source. Nevertheless, you are to make predictions based on the data.

The pairing is guaranteed to be consistent, i.e. each team will play against exactly one other team.

Input

The first line of input consists of an even integer N ($2 \le N \le 5000$).

$\frac{N}{2}$ lines follow. The i-th line consists of two space-separated integers A_i and B_i, meaning that team A_i will play against team B_i in the final fixture. Each of $1, 2, \ldots, N$ appears exactly once in $\{A_1, \ldots, A_{\frac{N}{2}}, B_1, \ldots, B_{\frac{N}{2}}\}$.

N lines follow. The i-th line consists of three space-separated integers P_i, S_i, and C_i, representing the points of, goals scored by, and goals conceded by team i ($0 \leq P_i \leq 30000$; $0 \leq S_i, C_i \leq 10^5$).

Output

Output N lines. On the i-th line, output the best final rank of team i, followed by the worst final rank of team i. Separate the two numbers by one space.

Examples

input

```
2
1 2
3 100000 0
0 0 100000
```

output

```
1 2
1 2
```

input

```
4
1 3
2 4
4 4 1
2 3 3
4 4 3
0 1 5
```

output

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

input

```
2
1 2
1 1 0
5 2 7
```

output

```
2 2
1 1
```

input

```
10
2 3
9 10
1 6
7 8
4 5
47 65 12
34 34 20
34 36 24
28 32 22
23 25 25
23 21 28
19 23 34
15 21 32
14 17 39
5 11 49
```

output

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

Note

Sample input 1 is an example of extreme scorelines. Team 2 can still take first place, for example by winning the final game $262144 - 131072$.

Sample input 3 is an example of inconsistent data.

Sample input 4 is from this season's Hong Kong Premier League. (It turns out that the ranking is unchanged after the final fixture.)

G Go

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

One day, GFRIEND members went to a Go club to perform their song and learn how to play Go.

To teach them the basic rules of Go, the master gave them N white stones and M black stones. He then ask them to use the white stones to surround the black stones on the standard 19x19 Go board. The arrangement must satisfy the followings:

- They must use all N white stones and all M black stones.
- All black stones must be "captured". In order to capture a group of black stones, it must be surrounded by white stones in all 4 directions without gaps and spaces.
- Though it is not necessary for the black stones to form a single group, all white stones must be necessary, i.e. removing any one white stone will cause at least one black stone not being captured.

The above rules can also be expressed by the followings:

- The 19x19 board must contain exactly N white stones and exactly M black stones.
- Each white stone must be directly adjacent to (4 directions) at least 1 black stone.
- Each black stone must be directly adjacent to (4 directions) 4 black or white stones.

Help GFRIEND by writing a program to find out if there is a possible arrangement, and if there is, output any one.

Input

The input consists of 2 integers N and M.

$1 \leq N, M \leq 80$.

Output

If there is no possible arrangement that satisfies the rules, output `Impossible`.

Otherwise, output a 19x19 grid that contains any valid arrangement. Use `.` to represent empty space, `o` to represent white stone and `x` to represent black stone.

Examples

input

```
20 25
```

output

```
...................
...................
...................
...................
...................
...................
...................
...................
...................
...................
..........ooooo....
.........oxxxxxo...
.........oxxxxxo...
.........oxxxxxo...
.........oxxxxxo...
.........oxxxxxo...
..........ooooo....
...................
...................
```

input

```
21 10
```

output

```
...................
...................
...................
...ooo.............
..oxxxo............
...ooxo............
.....o.............
.........o..o......
........oxooxo.....
........oxxxxo.....
.........oooo......
...................
...................
...................
...................
...................
...................
...................
...................
```

input

```
5 2
```

output

```
Impossible
```

H Handicap

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
第三屆 · 2018年5月22日

In Hackerland, sports programming is the most popular activity. More than half of its residents has a rating on Hackforces, the official programming competition platform there.

What is so unique about Hackforces is that 1-versus-1 matches are available! Two players can match online and enjoy a battle of wits anytime. The match can be boring though, if, for example, an absolute beginner with rating 1 fights a *Legendary Grandmaster* with rating 10^9 - everyone knows the outcome before the match starts.

Therefore, Hackforces has a *handicap system*, allowing the higher-rated player to play with some disadvantages, thus making the match more even. There are N types of handicaps, and the i-th one is equivalent to reducing one's rating by $D[i]$ points. Here are some of them:

- Type With One's Nose - Self-explanatory.
- You Only Submit Once - Only one submission attempt per task.
- Blindfolded - Code while wearing a blindfold.

Seeing that many strong programmers do not voluntarily select a suitable handicap, Hackforces would like to install an *auto-handicap* feature. Given the ratings of two players X and Y ($X \leq Y$), *auto-handicap* should automatically pick an index i between 1 and N (inclusive), such that $Y - D[i] > 0$ and $|Y - D[i] - X|$ is minimized. In other words, the new rating $Y_{new} := Y - D[i]$ of the second player should be positive and as close to X as possible.

There are three subtle details to keep in mind. First, if playing without handicap yields the smallest rating difference, then it is preferred to play without handicap. Second, if none of the handicaps can make Y_{new} positive, then playing without handicap is the only choice. Third, if two or more different valid handicaps yield the same rating difference, then the one with the smallest index is preferred.

You are to implement this feature to pick the best handicap for Q matches, according to the rules above.

Input

The first line of input consists of a single integer N.

The second line of input consists of N space-separated integers $D[1], D[2], \ldots, D[N]$.

The third line of input consists of a single integer Q.

For the next Q lines of input, the i-th line consists of a pair of integers X_i and Y_i, the ratings of the two players in the i-th match. It is guaranteed that $X_i \leq Y_i$.

For all test cases, $1 \leq N, Q \leq 10^5$, $1 \leq D[1] < D[2] < \cdots < D[N] < 10^9$, $1 \leq X_i \leq Y_i \leq 10^9$.

Output

Output N lines, the i-th line corresponding to the i-th match.

If the best choice for the i-th match is to play without handicap, output 0.

Otherwise, output the index corresponding to the chosen handicap.

Example

input

```
5
2 3 5 10 100
5
1 1
1500 1501
1500 1502
1500 1504
10 65
```

output

```
0
0
1
2
4
```

I Infection

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

Byteland is a country consisting of N cities, numbered from 1 to N, and they are connected by M bidirectional roads. It is possible to reach each city from any other. Each road connects two cities a and b. Two cities are neighbor if and only if they are directly connected by a road.

Scientists in Byteland found out that there will be an outbreak of Virus B in the coming weeks. The virus will first appear in one of the N cities, make it into an infected city. Then, on each day the virus will spread to exactly one uninfected city which is neighbor to an infected city. the The infection process will end when all N cities become infected.

As Virus B is a fatal and strong virus, they want to make some precautions against Virus B to minimize the impacts. Luckily, the scientists found out that Virus B has similar DNA structure with another Virus A, which appeared in Byteland 100 years ago, so the scientists predict that Virus B will infect different cities in a similar way as Virus A. More precisely, let the order of cities infected by Virus A be $A_1, A_2, \ldots, A_N$ and the order of cities infected by Virus B be $B_1, B_2, \ldots, B_N$, the scientists predict that B will be the lexicographically smallest ordering such that is also lexicographically greater than A.

An ordering $P_1, P_2, \ldots, P_N$ is considered lexicographically smaller than another ordering $Q_1, Q_2, \ldots, Q_N$, if $P_i < Q_i$, for the first i where P_i and Q_i differ.

Given the structure of Byteland and the infection log of Virus A, please help the scientists to predict the order of cities infected by Virus B or print -1 if such ordering does not exist.

Input

The first line contains two integers N and M, the number of cities and roads in Byteland $(1 \leq N, M \leq 10^5)$.

The following M line contains two integers u_i and v_i, describing the i^{th} road connecting city u_i and v_i $(1 \leq u_i, v_i \leq N)$.

The last line contains N integers A_i, describing the order of cities infected by Virus A 100 years ago.

It is guaranteed that it is possible to reach each city from any other in Byteland. Also, it is guaranteed that the given ordering must be a valid infection ordering.

Output

Output N integers in the first line, the predicted infection ordering of Virus B. If such ordering does not exist, output -1 in the first line.

Examples

input

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

output

```
3 1 4 2
```

input

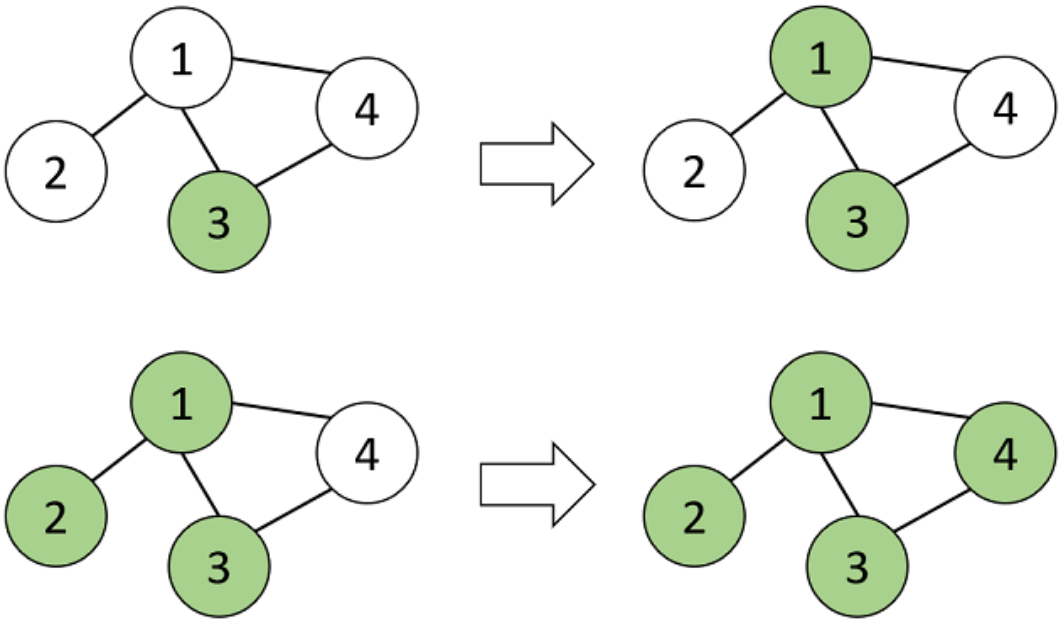
```
4 3
1 2
2 3
3 4
4 3 2 1
```

output

```
-1
```

Note

In the first sample, the order of cities infected by Virus A is as follow:



Therefore, the lexicographically smallest infection order for Virus B, which is lexicographically greater than infection order for Virus A, is $3, 1, 4, 2$

J Jakanda Forever

Time Limit: 2 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

In East Hackerland, there is a country called Jakanda which possesses highly advanced technology. It consists of N cities and $N-1$ bidirectional roads. Each road has its own length and connects two cities. It is possible to travel from a city to any other city via these roads.

The Black Panda, king of Jakanda, is facing a problem. Some of the cities are having riot. A city under riot would send out a rebel army and try to capture one other city ("targeted city"). Black Panda is not going to let it happen. He would send a scout from the targeted city to the city under riot. Both the rebel and scout will walk for 1 mile toward their own destination during the day and rest at night. The scout would only spot the rebel army only if they are at the same location and it is at night. If the scout fails to spot the army, the rebel army would capture its target city.

Other than the riot, the length of some of the roads in Jakanda would change due to damage of the war.

Now there are Q events in chronological order. Each event is one of the two types:

`1 u v`, which denote city u is now under riot and is sending a rebel army to capture city v. Note that city v might have rebellion or captured before, but in these cases we would consider Black Panda had already suppressed it.

`2 i l`, which denote the i^{th} road is now changed to length l.

For each type 1 event, output `JAKANDA FOREVER` if the scout could spot the rebels, output `WE NEED BLACK PANDA` otherwise.

Input

The first line contains a single integer N, the number of cities ($1 \leq N \leq 5 \times 10^5$).

The following $N-1$ lines contains the roads information, each line contains three integers: u_i, v_i and l_i, which means the i^{th} road connect city u_i and v_i with length of l_i miles ($1 \leq u_i, v_i \leq N$; $u_i \neq v_i$; $1 \leq l_i \leq 10^9$).

Then a single line is followed, which contains a single integer Q ($1 \leq Q \leq 5 \times 10^5$).

The following Q lines contains the event information, each line contains three integer. The first integer $type$ denote the event type.

For $type = 1$, two integers, `u v` is followed. u is the city under riot and v is the target city ($1 \leq u, v \leq N$; $u \neq v$).

For $type = 2$, two integers, `i l` is followed. It means the i^{th} road's length is changed to l miles.

Output

For each type 1 event, output `JAKANDA FOREVER` if the scout is going to spot the rebels, `WE NEED BLACK PANDA` otherwise.

Example

input

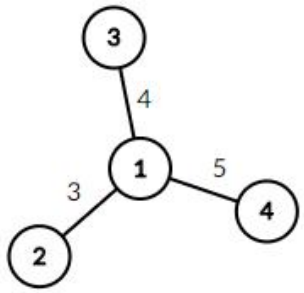
```
4
1 2 3
1 3 4
4 1 5
5
1 1 2
1 2 3
2 1 2
1 2 1
1 4 2
```

output

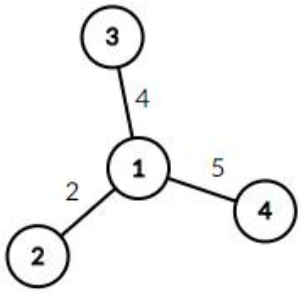
```
WE NEED BLACK PANDA
WE NEED BLACK PANDA
JAKANDA FOREVER
WE NEED BLACK PANDA
```

Note

The structure of Jakanda before the third event:



The structure of Jakanda after the third event:

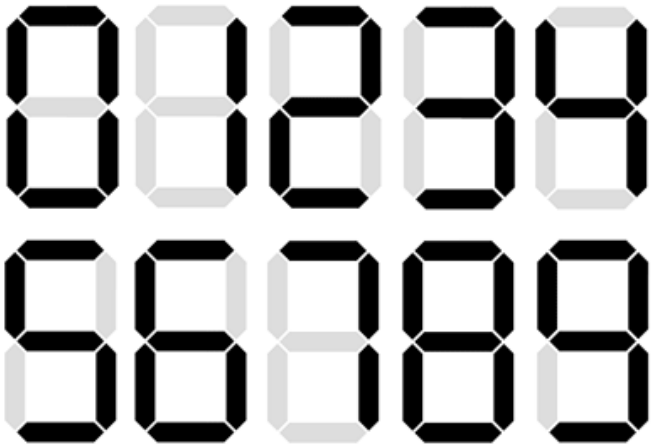


K Kids' Entertainment

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

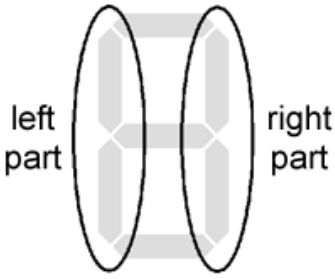
What do kids do when they go dine at a Chinese restaurant ("yum cha")? Back when portable game consoles and smartphones were not available, some kids may opt to play with toothpicks, forming figures on the dining table.

Hand-written digits contain curly strokes, but by using the seven-segment display, digits can be represented neatly using only straight segments. The diagram below shows digits 0 to 9 in seven-segment display:

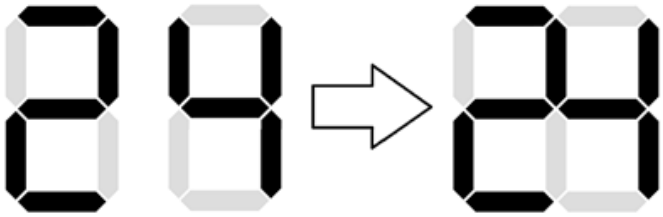


Alex wants to form a number between 0 and 99 (inclusive), by using toothpicks as segments in the seven-segment display (one toothpick per segment, naturally). He insists on forming exactly using two digits, so 0 will be written as 00, 1 will be written as 01, and so on.

Assume the tens digit is X and the ones digit is Y. To use fewer toothpicks, Alex invents a rule for *merging* X and Y. First, Alex defines the *left part* of a digit as the two vertical segments on its left. Similar for *right part*. Refer to the following diagram.



If the right part of X and the left part of Y are the same, then the parts can be merged, allowing a more compact and toothpick-friendly representation. See the following example.



There is one exception to this rule. As you may have noticed, if $X = 1$, then after merging it will look like one single digit. Therefore, Alex will not merge the two digits if the tens digit X equals 1.

Alex is trying to form T numbers. For each number, determine the number of toothpicks needed to form the number. Whenever digit-merging is possible, Alex will merge the digits.

Input

The first line of input consists of an integer T, the number of queries.

T lines follow. On the i-th line, there is an integer in the range $[0, 99]$, representing the i-th number that Alex wants to form. Leading zeroes will be added to integers smaller than 10, so that each line consists of two digits.

Other than the sample, your program will be judged on exactly one other test case. For that test case, $T = 100$.

Output

Output T lines. On the i-th line, output one integer, the number of toothpicks needed to form the i-th number.

Example

input

```
8
10
00
24
88
89
75
33
11
```

output

```
8
10
8
12
13
8
10
4
```

L Labyrinth

Time Limit: 1 second
Memory Limit: 256 MB
Input: standard input
Output: standard output

培正喇沙 編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE
第三屆 · 2018年5月22日

Hackerland's Theme Park has an exciting attraction: a labyrinth that can be treated as a $3 \times W$ grid. It has three special squares: $A = (1, a)$, $B = (3, b)$, and $X = (2, 1)$. Two players will start at A and B respectively and will try to reach X as quickly as possible. In one step, a person can move one square up, down, left, or right. It is possible for both of them to be located at the same square at the same time.

For convenience, let $dist(U, V)$ denote the distance between squares U and V, i.e. the number of steps needed to reach V from U. If V is not reachable from U, then $dist(U, V) = \infty$.

You are concerned that the game may be unfair, if $dist(A, X) \neq dist(B, X)$. Therefore, you are going to place some (possibly zero) obstacles at some of the squares, so that $dist(A, X) = dist(B, X) < \infty$.

Originally, the grid has no obstacles. You cannot add obstacles to squares A, B, and X. Find a way to create a fair labyrinth!

Input

The first and only line of input consists of three space-separated integers W, a, and b.

For all test cases, $1 \leq a, b \leq W \leq 100$.

Output

If there is no way of adding obstacles so that $dist(A, X) = dist(B, X) < \infty$, output `Impossible`.

Otherwise, output `Possible` followed by three lines. Output W characters on each of the next three lines. The j-th character of the i-th line should be:

- `A` (ASCII 65), if $(i, j) = A$;
- `B` (ASCII 66), if $(i, j) = B$;
- `X` (ASCII 88), if $(i, j) = X$;
- `*` (ASCII 42), if (i, j) is none of A, B, X and an obstacle is placed at (i, j);
- `.` (ASCII 46), if (i, j) is none of A, B, X and no obstacle is placed at (i, j).

If there are multiple solutions, output any one of them.

Examples

input

```
1 1 1
```

output

```
Possible
A
X
B
```

input

```
4 3 2
```

output

```
Impossible
```

input

```
3 3 1
```

output

```
Impossible
```

input

```
9 3 7
```

output

```
Possible
**A....**
X.**...**
.....*B**
```