Problem A
Passwords
Time limit: 5 seconds

Since 2011, the rules required for secure passwords have actually become more relaxed. The current set of rules is:

1. The password is between 9 and 20 characters in length inclusive (its length can be equal to 9 or 20).
2. The password contains only lowercase, uppercase, and non-alphanumeric characters. For this problem the only valid non-alphanumeric characters are: ! @ # $ % ^& * . , ; / ?
3. The password contains at least 2 lowercase letters
4. The password contains at least 2 uppercase letters
5. The password contains at least 1 number
6. The password contains at least 2 non-alphanumeric characters (as defined above)

All of your passwords from 2011 have expired, so you need to create new ones. You want to generate n new passwords for all your logins. You know that it is insecure for all of your passwords to be the same, so you require that **any pair of passwords of the same length must differ in at least 3 different positions**. For instance, if one of your passwords is `UVAhspc2024!!`, you cannot use the password `UVAhspc2024..` because they only differ at 2 positions. However, you could use `UvAhSpC2024!!` because it differs in 3 positions.

Input

Input consists of a single line containing the integer n ($1 \leq n \leq 2011$) - the number of passwords you must generate.

Output

Output n lines. Each line should contain a single password. Your list of passwords must satisfy the constraints in the statement.

Sample Input 1	Sample Output 1
2	UVAhspc2024!! UvAhSpC2024!!

Problem B
Pots of Gold
Time limit: 5 seconds

At long last, you've finally made it! You've found the end of the rainbow! You're not sure how it took people so long to find it–it was at $(0, 0)$ the whole time. Now that you've found it, you're ready to begin looking for pots of gold.

Unfortunately, there is no gold in the immediate vicinity. You see some large, circular metal pots in the distance, and you suspect there are even more further away. You don't have much time to collect gold, so you'll just have to pick a direction randomly and start walking in a straight line from $(0, 0)$, picking up all the gold in every pot you touch on the way.

On average, if you do pick a direction uniformly at random, how much gold will you get?

Input

The first line of input contains a single integer n, $(1 \leq n \leq 201200)$, the number of pots of gold. Each of the following n lines contains 4 integers x_i, y_i, r_i, v_i - the x and y coordinates for the center of the pot, the radius of the pot, and the amount of gold in the pot, $(|x_i|, |y_i| \leq 10^6, 1 \leq r_i \leq 10^6, 1 \leq v_i \leq 10^9)$.

It is guaranteed that no pot touches or contains $(0, 0)$. Due to leprechaun magic, pots may overlap with each other.

Output

Print one number, the expected sum of v_i after walking in a random direction as a decimal.

Your answer will be accepted if the absolute or relative error is at most 10^{-6}. That is, if the correct answer is y and your answer is x, your answer will be accepted if $\min\left(|x - y|, \frac{|x-y|}{y}\right) \leq 10^{-6}$.

Sample Input 1	Sample Output 1
1 1 1 1 10	2.5

Sample Input 2	Sample Output 2
3 4 -7 1 1 -3 -2 1 2 2 2 1 3	0.563575983426556

Problem C
Less Dice
Time limit: 1 second

You have a single fair n-sided die with faces numbered 1 through n. Find the expected (average) value of a single roll multiplied by two. It can be shown that this will always be a whole number.

Input

The first line contains one integer n, $(1 \leq n \leq 100000)$, the number of faces on the die.

Output

Print two times the expected value of a die roll.

Sample Input 1	Sample Output 1
2	3

Sample Input 2	Sample Output 2
5	6

Sample Input 3	Sample Output 3
100000	100001

Problem D
Battle
Time limit: 3 seconds

You the commander of the forces of Minas Tirith in the Battle of the Pelennor Fields. In order to keep your soldiers rested at all times, they have rotating shifts. Your current army consists of n soldiers each with strength a_i. In each of the next n minutes, the i-th soldier will be replaced by a new soldier with strength b_i in the i-th minute. The strength of your army is equal to the sum of all active soldier strengths. You want to find the maximum strength of your army over the next n minutes. This can include the strength of the army in the 0-th minute before any soldiers have been replaced.

Input

The first line will contain a single integer n ($1 \leq n \leq 2 \cdot 10^5$), the size of your army. The second line will contain n integers representing a, the strengths of the n soldiers in your initial army. The third line will contain n integers representing b, the strengths of the n soldiers that replace your army. ($1 \leq a_i, b_i \leq 1000$).

Output

Output a single integer, the maximum total strength of the army over the next n minutes.

Sample Input 1

```
5
1 1 5 5 5
5 5 1 1 1
```

Sample Output 1

```
25
```

Sample Input 2

```
5
5 5 5 1 1
1 1 1 5 5
```

Sample Output 2

```
17
```

Sample Input 3

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

Sample Output 3

```
61
```

Problem E
HULK NUMBER
Time limit: 3 seconds

HULK HAVE NUMBER n. HULK NEED SPLIT NUMBER n. HULK NEED TWO NUMBER $1 \leq x, y \leq n$ SUCH THAT $xy = n$. HULK WANT x, y CHOSEN SO THAT $x + y$ DIVISIBLE BY 3. HELP HULK!

Input

A single line containing the integer n, $1 \leq n \leq 10^{18}$.

Output

If it is possible to do so, output two positive integers $1 \leq x, y \leq n$ such that $xy = n$ and $x + y$ is divisible by 3. Otherwise, output -1.

Sample Input 1	Sample Output 1
6	-1

Sample Input 2	Sample Output 2
9	3 3

Sample Input 3	Sample Output 3
32	8 4

Sample Input 4	Sample Output 4
1000000000000000000	-1

Problem F
Breakfast Bidders
Time limit: 2 seconds

The beloved breakfast restaurant, the House Serving Pancakes and Coffee (HSPC), is looking for new kitchen equipment at a bargain price! Fortunately, they know just the place to look: Coff-E-Bay, a resale website just for breakfast restaurant owners looking to exchange their cookware and supplies.

On Coff-E-Bay, sellers can list items at a starting price of their choice, and buyers can place bids to buy listed items. Through a simple auction system, items listed on Coff-E-Bay are sold to the buyer who placed the highest bid. However, if a buyer's bid is less than the starting price, their bid is considered invalid and does not count. Your task is to write code for Coff-E-Bay's backend that will automate the selling process by identifying the buyer to whom any listed item should be sold.

Input

The first line of input will contain 2 positive integers: $1 \leq n \leq 3$, the total number of listings, followed by $1 \leq k \leq 2016$, the number of bids that have been placed per listing. n groups of $k + 1$ lines each follow. The first line of each contains the name of the item being listed, followed by its starting price (in dollars, with up to two digits after the decimal). The next k lines of each listing contain the username of a buyer, followed by the price (in dollars, with up to two digits after the decimal) that they bid for the current listing.

Each listing will have the same number of bids, no item will be listed twice, and each buyer can only place one bid per item. However, buyers are allowed to place bids for multiple listings if they wish.

All names will be strings of at most 45 letters. No listing or bid will be more than $10,000.00, and **you can assume that no two buyers will bid the exact same price in a single listing.**

Output

For each listing, print the name of the recipient. If no one bid high enough, print "Unsold".

Sample Input 1	Sample Output 1
2 3 hotplate 80.00 SunriseSyrupCafe 81.50 BrewBistro 90.00 EggsquisiteMornings 75.00 toaster 150.00 PancakeParadiseInn 130.00 WaffleWonderland 135.50 SunriseSyrupCafe 149.99	BrewBistro Unsold

Problem G
Platforms
Time limit: 5 seconds

Mario wants to complete a level that consists of n platforms. The i-th platform has a height of h_i. He starts on the leftmost platform and wants to jump to the rightmost platform. When Mario jumps, he will always jump to the right. As an expert jumper, Mario knows that he can move between 1 and k platforms to the right in a single jump. He cannot jump past the rightmost platform.

Mario knows that it is hard to jump large vertical distances, up or down, so he wants minimize the tallest jump (by absolute vertical distance) he makes while completing the level. Find this minimum.
Formally, choose some subsequence of h. Let's call the chosen indices of the subsequence, $s_1 = 1, s_2, \ldots, s_l = n$, where l is the length of the subsequence. You want to minimize the maximum of $|h_{s_i} - h_{s_{i-1}}|$ over all $2 \leq i \leq l$. Output this minimum value.

Input

The first line contains two integers n, k, $2 \leq n \leq 10^5, 1 \leq k \leq n-1$ - the number of platforms and maximum horizontal distance of a jump. The second line contains n integers representing h, the heights of the n platforms ($1 \leq h_i \leq 10^9$).

Output

Output the smallest maximum absolute vertical distance that Mario must jump in order to reach the rightmost platform.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

```
5 2
1 1000000000 1 1000000000 1
```

Sample Output 2

```
0
```

Sample Input 3

```
4 2
1 1000000000 1 1000000000
```

Sample Output 3

```
999999999
```

Problem H
Treedles
Time limit: 4 seconds

You may not know this, but Toodles has a huge underground building with n rooms where he stores all his tools.

The structure of the building is a tree. A tree is a connected graph with $n-1$ edges. Each node in the tree is a room. Toodles is very organized, so each room stores tools of a single type. The i-th room has x_i tools of type t_i. **Multiple different rooms may store tools of the same type**. Room 1, is the only room that exits to the outside world.

Today, Mickey requested every last one of Toodles' tools. To collect all of them, Toodles will go to each tool and carry it from its starting room to the exit in room 1, passing through other rooms on the way. Note that the path will be unique because the building is a tree.

Because Toodles is very organized, he has sensors in each room that detects tools of the same type as is stored in that room (t_i). The sensor is supposed to count the number of tools of type t_i that leave room i. The sensor in room 1 will count every tool of type t_1 since every tool will exit through room 1. Unfortunately, the sensors malfunctioned today.

Can you give Toodles the final counts for each of his sensors after all the tools have been delivered?

Input

The first line contains a single integer n, ($1 \leq n \leq 2 \cdot 10^5$) - the number of rooms. The second line contains n integers representing the array t, ($1 \leq t_i \leq n$). The third line contains n integers representing the array x, ($1 \leq x_i \leq 10^9$). The fourth line will contain $n-1$ integers. The i-th value will represent the parent of the $i+1$-th node in the tree structure of the building. The input is guaranteed to be a tree rooted at node 1.

Output

Output n integers. The i-th integer should be the number of tools of type t_i that exit node i as Toodles delivers every tool to Mickey.

Sample Input 1

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

Sample Output 1

```
5 8 2 3 4
```

Sample Input 2

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

Sample Output 2

```
25 12 12 4 12 6 7 8 9
```

Problem I
Roman Game
Time limit: 5 seconds

Julius Caesar is playing a game against a senator. Initially the game begins with n piles of stones. The i-th pile contains a_i stones. Players alternate turns with Caesar playing the first move. During their turn, a player will choose a pile and split that pile into 2 **or more** piles of any positive size. **This means that a pile with a single stone cannot be split**. The piles created from a split can be **different** sizes. For instance, you could split a pile of 5 stones into two piles of 1 stone and a pile of 3 stones. If a player cannot make a move, they lose. Output if Caesar or the senator will win the game if both players play optimally.

Input

The first line contains a single integer n, ($1 \leq n \leq 2 \cdot 10^5$) - the initial number of piles.
The second line will contain the array a of n integers representing the initial number of stones in each pile, ($1 \leq a_i \leq 10^9$).

Output

Output "CAESAR" if Caesar will win and "SENATOR" if the senator will win.

Sample Input 1	Sample Output 1
2 2 3	CAESAR

Sample Input 2	Sample Output 2
3 2 3 4	SENATOR

Sample Input 3	Sample Output 3
5 1 1 1 1 1	SENATOR

Problem J
Big Rooks
Time limit: 3 seconds

There is an infinite chessboard with n 3×3 "big" rooks on the board. A big rook can move any number of squares in a horizontal or vertical line. A big rook can capture another big rook if any square of the big rook would intersect with any square of another big rook after a single move. When a big rook captures another big rook, it will move to the exact location of the captured big rook and the captured big rook will be removed from the board.
Given the initial configuration of the n big rooks, calculate the minimum number of rooks that can remain after a sequence of only capturing moves. **Moves that do not capture a piece are not allowed**.

Input

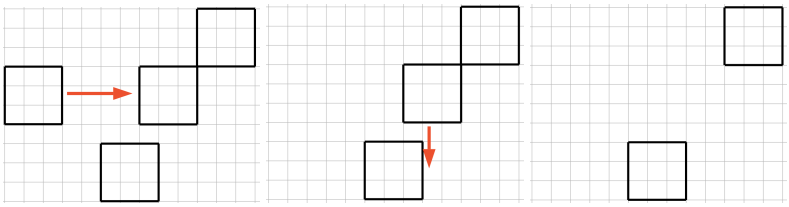
The first line contains a single integer n, ($1 \leq n \leq 2 \cdot 10^5$) - the number of big rooks in the initial state.
Each of the next n lines contains two integers x_i and y_i, ($1 \leq x_i, y_i \leq 2 \cdot 10^5$), the x and y coordinate of the **center** of the i-th big rook. No two big rooks will initially overlap in any square.

Output

Output the minimum number of big rooks that can remain after a series of only capturing moves.

Sample 1 Explanation

First rook 1 can capture rook 2. Rook 1 moves from (1,6) to (9,6) as it captures rook 2. Now rook 1 can capture rook 3 moving to (7,2). The two remaining rooks cannot attack each other, so no more moves can be made. It can be shown that there is no sequence of capturing moves that results in fewer than 2 remaining rooks.



Sample Input 1

```
4
2 6
9 6
7 2
12 9
```

Sample Output 1

```
2
```

Sample Input 2

```
3
1 1
8 4
5 7
```

Sample Output 2

```
3
```

Problem K
Squid Hopscotch
Time limit: 5 seconds

As you know, squids have n legs. A squid hopscotch board has n squares.

A squid starts by putting any number of their legs, including 0, onto the hopscotch board with each leg getting a starting square. Some of the squares will be left empty if the squid chooses not to place all of their legs. The game proceeds in hops. If a squid currently has a leg on square i, then they must hop and land with that leg on square p_i. Every single leg must make this transition during every hop.

Given these rules, sometimes two legs will land on the same square. However, squid hopscotch squares aren't big enough to fit more than one leg, so the squid would lose in this scenario.

As long as this never occurs, the squid can keep hopping forever. Sally the squid has played thousands of games of squid hopscotch so she now knows many different initial placements of her legs that will never cause her to lose. However, she still hasn't found all of them. Can you tell Sally how many initial placements of her legs allow her to play indefinitely without ever losing?

Two placements will be considered different if the set of their starting squares is different (**The specific assignment of leg index to square index does not matter**). Because the answer can be very large, output the answer modulo $10^9 + 7$.

Input

The first line contains a single integer n ($1 \leq n \leq 202100$). The second line contains n integers $1 \leq p_i \leq n$. p_i represents the square that a leg must move to after a hop if it starts on square i.

Output

Output the number of unique starting placements that allow Sally to play hopscotch indefinitely, modulo $10^9 + 7$.

Sample Input 1

```
4
2 1 1 2
```

Sample Output 1

```
9
```

Sample Input 2

```
4
1 1 1 1
```

Sample Output 2

```
5
```

Sample Input 3

```
9
4 8 9 1 6 9 8 6 1
```

Sample Output 3

```
30
```

Problem L
Secret Bits
Time limit: 5 seconds

You are a recruiter for the Hiding Spies Public Committee (HSPC). The Association for Cruel Machinations (ACM) are still up to no good. In fact, their numbers have been growing at an alarming rate. We were able to intercept an encrypted binary string, a string consisting of just 0 and 1, from the ACM that contains the number of new recruits this year. However, we don't know how to perfectly reverse their encryption scheme. We just know some series of the following two decryption operations will result in the original value. **Note that you can perform operation 2 without performing operation 1 and vice versa**.

1. Change a substring that is 00 to 11
2. Change a substring that is 01 to 10

You can't let the ACM get away with hiring more new recruits than you, so you must prepare for the worst. Over all possible original values (any possible series of decryption operations), what is the maximum number of new recruits the ACM could have?

Input

The first line of the input contains a single integer n, $(1 \leq n \leq 2 \cdot 10^5)$ - the length of the encrypted binary string. The second line of input contains the binary string s - the encrypted binary string.

Output

Output a single line containing the maximal binary value that can be created after any sequence of decryption.

Sample Input 1	Sample Output 1
3 101	110

Sample Input 2	Sample Output 2
3 001	111

Sample Input 3	Sample Output 3
7 1001101	1111110

Problem M
MIS
Time limit: 5 seconds

The task of setting problems for a "time capsule"-themed programming contest is not an easy one, especially when the contest in question has a history of only making sets with strictly increasing problem difficulties. The lazy coordinator just breaks tradition, but you're better.

You have a list of n proposed problems ordered chronologically by the unique previous year of the contest that they reference. Each problem has a unique difficulty a_i. A valid problemset is a (not necessarily proper) subset of the problems that is increasing both in year of reference and difficulty. A *maximal* valid problemset is a valid problemset that is not a subset of any other valid problemset.

For each k from 1 to n, compute the number of maximal valid problemsets of length k.

Input

On the first line is a single integer $1 \leq n \leq 2\,500$, the number of proposed problems.

On the second line are n integers $1 \leq a_i \leq n$, $a_i \neq a_j$ for $i \neq j$, the difficulties of the problems when ordered by year.

Output

Output n space-separated integers, the k^{th} of which is the number of maximal valid problemsets of length k. Since these values can be large, compute them modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
3 1 2 3	0 0 1

Sample Input 2	Sample Output 2
3 3 1 2	1 1 0

Problem N
2020
Time limit: 5 seconds

You are given n 2-d integer points. $(0, 0)$ and $(2020, 0)$ will always be in the input. You start at $(0, 0)$ and want to find the shortest path that allows you to reach $(2020, 0)$. If you are at point u, you can move to any point v as long as no other input point exists on the line segment from u to v. Output any shortest path from $(0, 0)$ to $(2020, 0)$.

Input

The first line of input contains the integer n, $(2 \leq n \leq 2 \cdot 10^5)$ - the number of visitable points. The next n lines each contain x_i and y_i, $(|x_i|, |y_i| \leq 2020)$, the coordinates of the $i - th$ point. The first two input points will always be $(0, 0)$ and $(2020, 0)$ and all points will be distinct.

Output

On the first line, output the integer k, the minimum number of points that must be visited to reach $(2020, 0)$. k should include $(0, 0)$ and $(2020, 0)$.
On the next k lines output the k coordinates on the path from $(0, 0)$ to $(2020, 0)$ in the order they are visited.

Sample Input 1

```
6
0 0
2020 0
5 0
4 -6
7 -3
2 -3
```

Sample Output 1

```
3
0 0
7 -3
2020 0
```

Sample Input 2

```
6
0 0
2020 0
-5 0
10 0
100 0
1000 0
```

Sample Output 2

```
5
0 0
10 0
100 0
1000 0
2020 0
```

Sample Input 3

```
4
0 0
2020 0
1 1
1 -1
```

Sample Output 3

```
2
0 0
2020 0
```