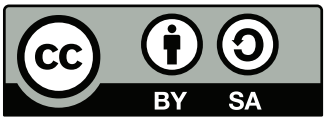
BAPC 2019

The 2019 Benelux Algorithm Programming Contest

Problems

A	Appeal to the Audience
B	Breaking Branches
C	Conveyor Belts
D	Deck Randomisation
E	Efficient Exchange
F	Find my Family
G	Gluttonous Goop
H	Historic Exhibition
I	Inquiry II
J	Jazz it Up!
K	Keep Him Inside
L	Lucky Draw



A Appeal to the Audience

Time limit: 1s

You are the director of the upcoming Bordfite Arena Progaming Competition. You have invited a bunch of players and are now setting up the matches for the knockout tournament that will determine the winner. As you may know, Bordfite Arena is a game that heavily rewards skill and very little luck is involved. This implies that whenever any number of players play a game of Bordfite Arena, the most skilled player will always win! Hence the winner of the tournament is already known, and you are a bit worried about this. How will you appease the audience?

You embark on a short quest to find out what the audience finds interesting. No surprises there: people find it most interesting when they see skillful players compete. Whenever a match is played, the happiness the audience gets from a match is the sum of the skill levels of the players. The total happiness the audience gets from the tournament is the sum of the happiness obtained during all matches. This is very useful information, because of course you want the audience to be as happy as possible at the end of the tournament.

Moreover, you invested some time to ask people what kind of knockout format they like. It turns out that instead of the plain old binary tree for the knockout schedule, they prefer a specific weird-looking rooted tree, and so you decide to use that. This means the final step for you to complete is to divide the players over the leaves of the given tree so that over the entire tournament, the happiness of the audience is maximized.

Input

- The first line contains integers $3 \leq n \leq 10^5$ and $1 \leq k \leq n-1$, the number of nodes of the tree and the number of players. The nodes are labelled 0 through $n-1$, and 0 is the root of the tree.
- The second line contains k integers $0 \leq a_1, \ldots, a_k \leq 10^9$, denoting the skill values of the players.
- Then follow $n-1$ lines, the ith of which ($1 \leq i \leq n-1$) contains the parent $0 \leq p_i < i$ of node i.

It is guaranteed that the tree has exactly k leaves and that there are no nodes with exactly one child.

Output

- Output the maximal possible happiness the audience can obtain from this tournament.

Sample Input 1

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

Sample Output 1

```
17
```

Sample Input 2

```
11 7
30 5 15 1 3 100 50
0
0
1
0
2
5
2
5
5
1
```

Sample Output 2

```
454
```

B Breaking Branches

Time limit: 1s

CC-BY 2.0 By DymphieH on Flickr

Your parents decided that it would be "fun" to spend the entire Sunday walking near the Mookerheide close to Nijmegen.

Although you can pass the time by solving programming problems in your head, your siblings do not have the same luxury. After a short while, your younger sister Alice and your big brother Bob find themselves hopelessly bored. Together, they try to figure out if they can pass the time with a game (a problem that would later be referred to as the Bob and Alice Pastime Conundrum). Finally, they come up with the following simple game.

They find a single branch of length n that will be the main object of the game. Alternatingly, Alice and Bob choose a piece of branch and break it into two parts, in such a way that both parts have integer lengths. The last player who is able to break one of the pieces wins. Alice gets to start, as she is the younger of the two.

Of course, you already have the game figured out in your head. Assuming Bob plays optimally, can Alice win the game? And if so, what move should she make first?

Input

- A line containing a single integer $2 \leq n \leq 10^9$, the length of the branch.

Output

- On the first line print the name of the person who wins, `Alice` or `Bob`.
- If Alice can win, print the length of a piece of branch Alice can break off as a winning move. This should be an integer between 1 and $n - 1$, inclusive.

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

Sample Input 1	Sample Output 1
2	Alice 1

Sample Input 2	Sample Output 2
3	Bob

C Conveyor Belts

Time limit: 1s

You are an employee of the Boxing And Processing Company and you are tasked with distributing boxes in one of the company's enormous warehouses. At BAPC Ltd., boxes travel by conveyor belt. Two conveyor belts can be merged into one by letting one drop its content onto the other. On the other hand, a belt can be split into two by using a splitter, which sends a specific portion of its input to its first output and the rest to its second output. Normally your splitters are adjustable, being able to distribute its input over its output belts in any conceivable ratio, but in an attempt to cut costs your boss has ordered cheap knock-off splitters which can only distribute its input in a fixed ratio $a : b$. Instead of arguing with your boss about how you really need some $c : d$ splitters, you decide to make them yourself.

Of course, with your frugal boss, you have to watch your costs. You cannot build your splitter system in such a way that there are boxes that never leave the system. Nor can you use splitters that never receive any boxes. Finally, you cannot use too many $a : b$ splitters in total.

Given the ratios $a : b$ and $c : d$, construct a network of belts and at most 200 knock-off $a : b$ splitters that has a single global input belt and two global output belts over which the global input is distributed in a ratio $c : d$.

Note that a splitter of ratio $x : y$ sends exactly $\frac{x}{x+y}$ of the incoming boxes to the first output and exactly $\frac{y}{x+y}$ of them to the second output.

Input

- The first line contains two integers $1 \leq a, b \leq 10^9$ with $a + b \leq 10^9$ denoting the ratio $a : b$ of your knock-off splitter.
- The second line contains two integers $1 \leq c, d \leq 10^9$ with $c + d \leq 10^9$ denoting the ratio $c : d$ of your desired splitter.

Output

- The first line contains an integer $1 \leq n \leq 200$, the number of $a : b$ splitters used.
- Then follow n lines, the i-th of which contains two integers $-2 \leq l_i, r_i < n$.

Here l_i is the index of the splitter connected to the left output of the i-th splitter, where it deposits $a/(a + b)$ of its input. Similarly r_i is the index of the splitter receiving $b/(a + b)$ of the input of the i-th splitter. The splitters are indexed starting from 0. The global input is connected to splitter 0. The special values -1 and -2 for l_i and r_i correspond to the first

and second global output, which need to receive $c/(c+d)$ and $d/(c+d)$ of the global input respectively.

Note that you cannot place a splitter in such a way that no box ever reaches it, nor in such a way that a box that passes through it will never reach the output.

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

Sample Input 1	**Sample Output 1**
2 3 3 2	1 -2 -1

Sample Input 2	**Sample Output 2**
1 2 3 4	3 -1 1 2 1 0 -2

Sample Input 3	**Sample Output 3**
1 2 1 2	3 -2 1 2 0 1 -1

D Deck Randomisation

Time limit: 1s

Alice and Bob love playing Don'tminion, which typically involves a lot of shuffling of decks of different sizes. Because they play so often, they are not only very quick at shuffling, but also very consistent. Each time Alice shuffles her deck, her cards get permuted in the same way, just like Bob always permutes his cards the same way when he shuffles them. This isn't good for playing games, but raises an interesting question.

They know that if they take turns shuffling, then at some point the deck will end up ordered in the same way as when they started. Alice shuffles once first, then Bob shuffles once, then Alice shuffles again, et cetera. They start with a sorted deck. What they do not know, however, is how many shuffles it will take before the deck is sorted again.

Can you help them compute how many shuffles it will take? As Alice and Bob can only do 10^{12} shuffles in the limited time they have, any number strictly larger than this should be returned as `huge` instead.

Input

- The first line contains a single integer $1 \leq n \leq 10^5$, the number of cards in the deck.
- The second line contains n distinct integers $1 \leq a_1, a_2, \ldots, a_n \leq n$, where a_i is the new position of the card previously at position i when Alice shuffles the deck.
- The third line contains n distinct integers $1 \leq b_1, b_2, \ldots, b_n \leq n$, where b_i is the new position of the card previously at position i when Bob shuffles the deck.

Output

- Output a single positive integer $m > 0$, the minimal number of shuffles required to sort the deck, or `huge` when this number is strictly larger than 10^{12}.

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

Sample Input 2	Sample Output 2
6 5 1 6 3 2 4 4 6 5 1 3 2	5

Sample Input 3	**Sample Output 3**
8 1 4 2 6 7 8 5 3 3 6 8 4 7 1 5 2	10

E Efficient Exchange

Time limit: 3s

You have recently acquired a new job at the Bank for Acquiring Peculiar Currencies. Here people can make payments, and deposit or withdraw money in all kinds of strange currencies. At your first day on the job you help a customer from Nijmegia, a small insignificant country famous for its enormous coins with values equal to powers of 10, that is, $1, 10, 100, 1000$, etc. This customer wants to make a rather large payment, and you are not looking forward to the prospect of carrying all those coins to and from the vault.

You therefore decide to think things over first. You have an enormous supply of Nijmegian coins in reserve, as does the customer (most citizens from Nijmegia are extremely strong). You now want to minimize the total number of coins that are exchanged, in either direction, to make the exact payment the customer has to make.

For example, if the customer wants to pay 83 coins there are many ways to make the exchange. Here are three possibilities:

Option 1. The customer pays 8 coins of value 10, and 3 coins of value 1. This requires exchanging $8 + 3 = 11$ coins.

Option 2. The customer pays a coin of value 100, and you return a coin of value 10, and 7 coins of value 1. This requires exchanging $1 + 1 + 7 = 9$ coins.

Option 3. The customer pays a coin of value 100, and 3 coins of value 1. You return 2 coins of value 10. This requires exchanging $1 + 3 + 2 = 6$ coins.

It turns out the last way of doing it requires the least coins possible.

Input

- A single integer $0 \leq n < 10^{1000}$, the amount the customer from Nijmegia has to pay.

Output

- Output the minimum number of coins that have to be exchanged to make the required payment.

Sample Input 1	Sample Output 1
83	6

Sample Input 2	Sample Output 2
13	4

Sample Input 3	Sample Output 3
0	0

Sample Input 4	Sample Output 4
12345678987654321	42

F Find my Family

Time limit: 7s

CC-BY 2.0 By Ivan on Flickr

You are looking for a particular family photo with you and your favorite relatives Alice and Bob. Each family photo contains a line-up of n people. On the photo you're looking for, you remember that Alice, who is taller than you, was somewhere on your left from the perspective of the photographer. Also, Bob who is taller than both you and Alice, was standing somewhere on your right.

Since you have a large number of family photos, you want to use your computer to assist in finding the photo. Many of the photos are quite blurry, so facial recognition has proven ineffective. Luckily, the Batch Apex Photo Classifier, which detects each person in a photo and outputs the sequence of their (distinct) heights in pixels, has produced excellent results. Given this sequence of heights for k photos, determine which of these photos could potentially be the photo you're looking for.

Input

- The first line contains $1 \leq k \leq 1000$, the number of photos you have to process.
- Then follow two lines for each photo.
  - The first line contains a single integer $3 \leq n \leq 3 \cdot 10^5$, the number of people on this photo.
  - The second line contains n distinct integers $1 \leq h_1, \ldots, h_n \leq 10^9$, the heights of the people in the photo, from left to right.

It is guaranteed that the total number of people in all photos is at most $3 \cdot 10^5$.

Output

- On the first line, output the number of photos k that need further investigation.
- Then print k lines each containing a single integer $1 \leq a_i \leq n$, the sorted indices of the photos you need to look at.

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

Sample Input 2	**Sample Output 2**
4 4 140 157 160 193 5 15 24 38 9 30 6 36 12 24 29 23 15 6 170 230 320 180 250 210	2 2 4

G Gluttonous Goop

Time limit: 3s

As a prominent researcher in the laboratorium for Breathtaking Agriculture through Petri dish Cultivation, you keep on looking for new organisms that might be the food of the future. Recently you have discovered a new fungus-type organism that seems to be nutritious and very efficient in converting energy from food to body mass. You have placed a small batch surrounded by food in a petri dish and watched it grow for a bit.

However, now that the weekend has arrived, you would rather spend some time with your loved ones than stare at the contents of this petri dish all the time (even though it is a *fun guy*). You cannot leave without taking the necessary precautions. What if the fungus grows too large and starts eating away at the rest of the laboratory?!

You model the situation as follows: you divide the plane into 1×1-squares, and draw where the fungus currently is. You know that every time step, if the fungus occupies a square, it will expand to all eight neighbouring squares (and still occupy the initial square). You know how many time steps you will be gone for over the weekend, and now you want to know how many squares the fungus will occupy when you get back.

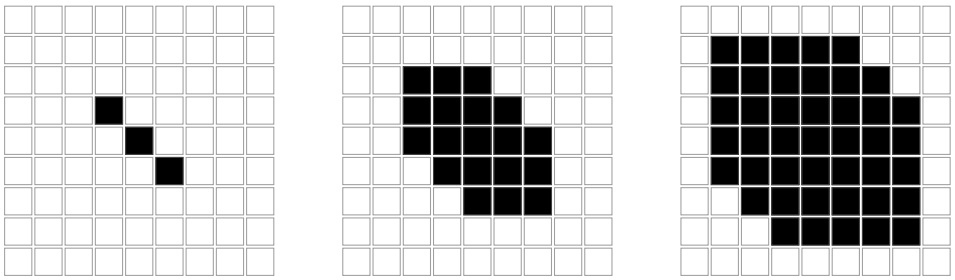


Figure G.1: Example of fungus growth: the fungus from sample 2 after 0, 1, 2 time steps. The middle image corresponds to the correct output for sample 2.

N.B.: In the input, the fungus will be given on a finite grid, but it can (and will!) grow beyond these boundaries. The fungus is not so easily contained.

Input

- First a line containing integers $1 \leq r, c \leq 20$, and $0 \leq k \leq 10^6$, denoting the number of rows and columns of the initial grid and the number of time steps.
- Then follow r lines of c characters, each character being either '`.`' or '`#`'. A '`#`' denotes that the fungus is occupying this square. The fungus need not be connected.

Output

- Output the number of squares the fungus occupies after k time steps have passed.

Sample Input 1

```
5 5 3
.....
.###.
.#.#.
.###.
.....
```

Sample Output 1

```
81
```

Sample Input 2

```
3 3 1
#..
.#.
..#
```

Sample Output 2

```
19
```

Sample Input 3

```
4 6 3
..##..
.#..#.
.#..#.
..##..
```

Sample Output 3

```
96
```

Sample Input 4

```
1 1 1000000
#
```

Sample Output 4

```
4000004000001
```

H Historic Exhibition

Time limit: 1s

The Benelux Artistic Pottery Consortium is preparing for an exhibit of its most prized urns and vases at a gallery in Nijmegen. Due to the sheer number of vases to be put on display the gallery has trouble finding a pedestal of the right size for every single vase. They have pedestals available that can either be placed normally or upside down and can be characterised by the diameter of their top and bottom surface. Moreover, the diameter of the top and bottom varies by at most one unit length.

For artistic reasons, it is important that the diameter of the base of a vase matches the diameter of the surface of the pedestal it is placed on. You have been asked to find a way to place all the vases on available pedestals. In order to make this work, you might need to turn some of the pedestals upside down. For example, Figure H.1 shows a possible assignment of pedestals to vases for sample input 1. Assist the gallery by writing a program to compute such an assignment.

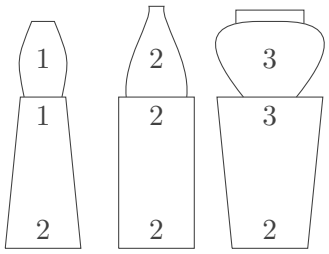


Figure H.1: Solution for sample input 1.

Input

- The first line contains two integers $0 \leq p, v \leq 10^4$ the number of pedestals and the number of vases.
- Then follow p lines, the i-th of which contains two integers $1 \leq a_i, b_i \leq 10^4$ denoting the diameters of the different sides of pedestal i. It is given that $|a_i - b_i| \leq 1$.
- Then follows a single line containing v integers $1 \leq c_1, \ldots, c_v \leq 10^4$, where c_i denotes the diameter of vase i.

Output

- Output v distinct integers $1 \leq x_1, \ldots, x_v \leq p$ such that vase i can stand on pedestal x_i, or print `impossible` if no assignment of vases to pedestals exists.

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

Sample Input 1

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

Sample Output 1

```
1
4
3
```

Sample Input 2

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

Sample Output 2

```
impossible
```

Sample Input 3

```
2 3
9 8
4 5
4 9 5
```

Sample Output 3

```
impossible
```

I Inquiry II

Time limit: 5s

For an undirected, simple graph $G = (V, E)$ we call a subset $V' \subseteq V$ an *independent set* if no two elements of V' are connected by an edge. An independent set of G is called a *maximum independent set* if there is no independent set in G with strictly more vertices. Given a specific kind of connected graph G, find the size of a maximum independent set of G.

Input

- The input starts with one line, containing integers n ($1 \leq n \leq 100$), the number of vertices in the graph, and m ($n - 1 \leq m \leq n + 15$), the number of edges in the graph.
- Then follow m lines, each containing integers a, b ($1 \leq a, b \leq n$) indicating that there is an edge between vertices a and b.

The graph given by this input is guaranteed to be both simple and connected: there is at most one edge between each pair of vertices, there are no loops, and there is a path between each pair of vertices.

Output

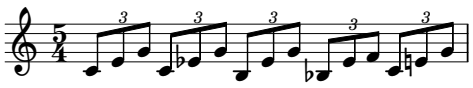
- Output the number of vertices in a maximum independent set of the input graph.

Sample Input 1	Sample Output 1
2 1 1 2	1

Sample Input 2	Sample Output 2
4 5 1 2 2 3 3 4 4 1 1 3	2

J Jazz it Up!

Time limit: 1s



Along with some friends you formed the Band of Atonal Percussionists and Cellists. You have been playing for some years together, but you feel unsatisfied with the current level of play. Doing research into some interesting new styles, you are gripped by the intricate details of the world of jazz.

While of course you cannot apply all the new things you have learned immediately, you want to start with improvising some nice new rhythmic figures in the music your band plays. You will play a rhythm where every bar has n beats in it, but then you split up every beat into m notes. In total, you will have nm notes per bar.

Everyone in the band knows that there is *no room for squares* in jazz. So the number of notes in a bar should be squarefree. That is, there is no number $k > 1$ such that k^2 divides the number of notes in a bar.

The percussionist has already suggested a number of beats per bar n; now it is up to you to find a number of notes per beat that does not leave any room for squares.

In the second sample we have $n = 30$ and $m = 7$. This works because $2 \leq m < n$ and $m \times n = 210$ has no divisor k^2 for any $k > 1$.

Input

- The input is a single squarefree integer $3 \leq n \leq 10^5$.

Output

- Output an integer $2 \leq m < n$ such that $m \times n$ is still squarefree.

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

Sample Input 1	Sample Output 1
3	2

Sample Input 2	Sample Output 2
30	7

Sample Input 3	Sample Output 3
13	10

K Keep Him Inside

Time limit: 1s

As a result of a long-standing war between the Sorcerers and the Orcs, you have been assigned as officer of one of the prison blocks. Recently the leader of the Orcs has been captured and placed inside a special cell. It works as follows: the cell is a convex polygon with at every vertex a guard tower in which a Sorcerer is placed.

Thanks to the recent agreement between the Sorcerers and Orcs, called the Beneficial Activities for Prisoners in Cells, the leader of the Orcs should be able to move around freely in his cell. You do not want your prisoner to escape, so you order the sorcerers to work together on a containment spell. If done right, this creates a magical aura around the prisoner that will prevent him from escaping.

In order for the containment spell to work, all Sorcerers need to channel a certain fraction of their maximum power into the spell such that two things hold:

- The spell needs to be perfectly balanced: the sum of the fractions of power over all Sorcerers must equal 1.
- The centre of the spell should coincide with the prisoner. This means that the average of the positions of Sorcerers, weighted by the fraction of power they are channeling, should be the location of the prisoner.

Given the layout of the cell and the location of the prisoner, assign a fraction of power each Sorcerer should spend so that the containment spell works.

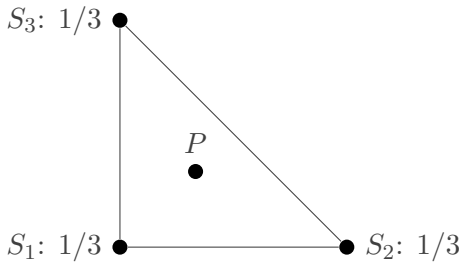


Figure K.1: The prison of sample 1. S_1, S_2, and S_3 are the Sorcerers, while P is the prisoner. Note that $\frac{1}{3}S_1 + \frac{1}{3}S_2 + \frac{1}{3}S_3 = \frac{1}{3}(0,0) + \frac{1}{3}(3,0) + \frac{1}{3}(0,3) = (1,1) = P$.

Input

- The first line contains $3 \leq n \leq 10$, the number of Sorcerers in guard towers and two integers $-10^4 \leq x, y \leq 10^4$, the coordinates of the prisoner.
- Then follow n lines, each of which contains two integers $-10^4 \leq x, y \leq 10^4$, the coordinates of a Sorcerer.

It is guaranteed that the locations are given in counter clockwise order and form a strictly convex polygon, i.e. no three points lie on a line. The prisoner is located strictly inside the polygon.

Output

- Output n lines where the ith line contains a floating point number between 0 and 1 inclusive: the fraction of power that the ith Sorcerer should use for the containment spell to work.

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

Your answer will be correct if the sum of fractions differs at most 10^{-4} from 1 and the weighted centre of your spell is within distance 10^{-4} of the prisoner. Note that it may not be sufficient to print your answer itself with 10^{-4} precision.

Sample Input 1	Sample Output 1
3 1 1 0 0 3 0 0 3	0.333333333333 0.333333333333 0.333333333333

Sample Input 2	Sample Output 2
4 2 1 0 0 4 0 4 4 0 4	0.5 0.25 0.25 0

Sample Input 3	Sample Output 3
5 4 3 0 2 0 -1 5 -2 5 4 2 5	0.2 0 0.1 0.7 0

L Lucky Draw

Time limit: 2s

You and your friends at the Betting against All Probability Club are visiting a casino where the following game is played.

Each of the n players starts with k lives and puts in a fixed amount of money. In each round of the game, each player flips a biased coin and loses a life if she gets tails. The game ends when only one player remains, in which case this person wins, or ends in a draw if all remaining players lose their last life in the same round. If there is a winner, she wins n times her original bet. In case of a draw, no one wins anything.

Being a BAPC member you quickly realize the casino has an edge here: whenever the game ends in a draw all of the contestants lose the money they bet. You are now wondering what exactly is the probability that this game ends in a draw, so you can figure out how much the casino profits on average.

Input

- One line containing two integers, $2 \leq n \leq 50$, the number of players, $1 \leq k \leq 50$, the number of lives each player has, and a floating point number $0.1 \leq p \leq 0.9$, the probability the coin lands heads.

Output

- Output a single floating point number: the probability of the game ending in a draw. Your answer should have an absolute error of at most 10^{-6}.

Sample Input 1	Sample Output 1
2 2 0.5	0.185185185

Sample Input 2	Sample Output 2
2 2 0.8	0.056241426

Sample Input 3	Sample Output 3
5 3 0.85	0.045463964