Northwestern European Regional Contest 2018

NWERC 2018

Eindhoven, November 25

Problems

Do not open before the contest has started.

This page is intentionally left (almost) blank.

Problem A
Access Points

A well-known programming contest is considering a new way to position its teams. For the contest all n teams have to be assigned some position (x, y) in an infinitely-large gym hall. To keep a good overview of the teams the following strategy is chosen:

All teams have been assigned a unique integer ID in the range $[1, n]$. Any two teams with IDs i and j, where $i < j$, must be placed at positions (x_i, y_i), (x_j, y_j), such that $x_i \leq x_j$ and $y_i \leq y_j$.

Unfortunately, someone already assigned the (fixed) internet access point for each team. The access points are quite big and only have one port, so access points for different teams are located at different positions. Every team must be connected to its designated access point by a direct UTP cable. The cost of a UTP cable of length ℓ is ℓ^2.

Find a placement for all teams, such that their respective order along both axes is maintained and the total cost of the required UTP cables is minimised. As the judges are not too worried about privacy, they are fine with two (or more) teams being placed at the exact same location or being arbitrarily close together. See Figure A.1 for an example.

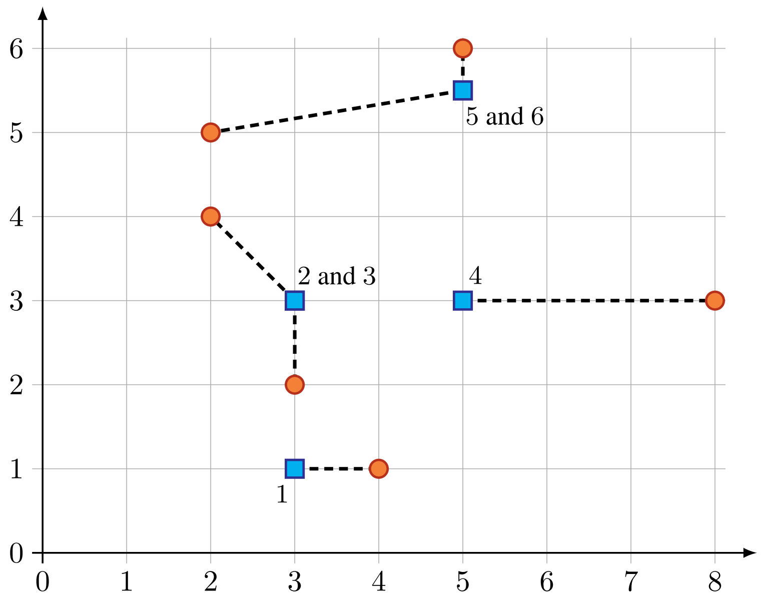


Figure A.1: Illustration of an optimal placement for Sample Input 1. Team placement (boxes), access points (circles), and required UTP cables (dashed).

Input

The input consists of:

- One line with one integer n ($1 \leq n \leq 10^5$), the number of teams.
- n lines, the ith of which contains two integers s_i, t_i ($1 \leq s_i, t_i \leq 10^6$), the location of the internet access point of team i.

No two access points are at the same position.

Output

Output the minimum total cost of all UTP cables required to connect the teams to their access points in an optimal legal layout.

Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1

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

Sample Output 1

```
22.5
```

Sample Input 2

```
6
11 6
23 7
24 11
24 32
27 38
42 42
```

Sample Output 2

```
0
```

Problem B
Brexit Negotiations

As we all know, Brexit negotiations are on their way—but we still do not know whether they will actually finish in time.

The negotiations will take place topic-by-topic. To organise the negotiations in the most effective way, the topics will all be discussed and finalised in separate meetings, one meeting at a time.

This system exists partly because there are (non-cyclic) dependencies between some topics: for example, one cannot have a meaningful talk about tariffs before deciding upon the customs union. The EU can decide on any order in which to negotiate the topics, as long as the mentioned dependencies are respected and all topics are covered.

Each of the topics will be discussed at length using every available piece of data, including key results from past meetings. At the start of each meeting, the delegates will take one extra minute for each of the meetings that has already happened by that point, even unrelated ones, to recap the discussions and understand how their conclusions were reached. See Figure B.1 for an example.

Nobody likes long meetings. The EU would like you to help order the meetings in a way such that the longest meeting takes as little time as possible.

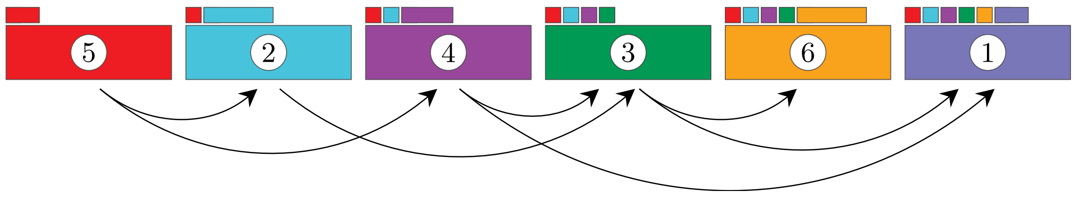


Figure B.1: Illustration of how time is spent in each meeting in a solution to Sample Input 2.

Input

The input consists of:

- One line containing an integer n ($1 \leq n \leq 4 \cdot 10^5$), the number of topics to be discussed. The topics are numbered from 1 to n.
- n lines, describing the negotiation topics.

  The ith such line starts with two integers e_i and d_i ($1 \leq e_i \leq 10^6$, $0 \leq d_i < n$), the number of minutes needed to reach a conclusion on topic i and the number of other specific topics that must be dealt with before topic i can be discussed.

  The remainder of the line has d_i distinct integers $b_{i,1}, \ldots, b_{i,d_i}$ ($1 \leq b_{i,j} \leq n$ and $b_{i,j} \neq i$ for each j), the list of topics that need to be completed before topic i.

It is guaranteed that there are no cycles in the topic dependencies, and that the sum of d_i over all topics is at most $4 \cdot 10^5$.

Output

Output the minimum possible length of the longest of all meetings, if meetings are arranged optimally according to the above rules.

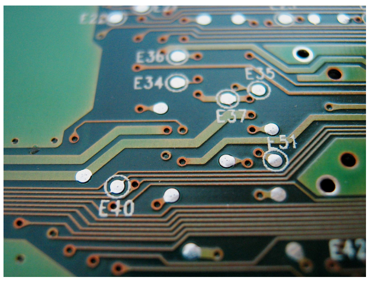
Sample Input 1	Sample Output 1

3
10 0
10 0
10 0
	12

Sample Input 2	Sample Output 2

6
2 2 4 3
4 1 5
1 2 2 4
3 1 5
2 0
4 1 3
	8

Problem C
Circuit Board Design



A circuit board. Public domain.

You have been hired at the *Nano Wiring Efficient Route Company* (NWERC) to help with the design of their new circuit boards. The circuits themselves have already been designed, and your task is to come up with a way to print them onto the blank boards that the company has bought.

More specifically, each circuit design consists of a number of connection points with some connections between them such that the resulting graph is connected and does not have any cycles (i.e., the graph is a tree).

You are free to place the connection points anywhere on the circuit board and solder the connections between them so that no two connections intersect (except at the connection points). The boards you ordered are fairly large, so there is no danger of running out of space. You can solder so precisely that connections and connection points can be considered infinitesimal.

This would all be very easy, however your boss persists that each connection needs to be a straight line of length exactly 1 mm (this is, so he says, to make sure the electrons do not have to travel around corners, which would be detrimental to the efficiency of the design).

You soon realise that battling with him will be unsuccessful. Your quickest way out of this is to etch a new design according to his specifications.

Input

The input consists of:

- One line with one integer n ($2 \leq n \leq 1\,000$), the number of connection points. The points are numbered from 1 to n.
- $n - 1$ lines, each with two integers a and b ($1 \leq a, b \leq n$), describing a connection between a and b.

It is guaranteed that these edges describe a valid tree.

Output

Output n lines, the ith of which contains two real numbers x_i, y_i, the coordinates of point i. To make the production feasible, the following restrictions apply:

- The distance between each pair of points should be at least 10^{-4}.
- The length of each edge should be 1, up to an *absolute* error of at most 10^{-6}.
- Edges that are not incident to the same vertex should be at least a distance 10^{-6} apart.
- The coordinates may not exceed an absolute value of $3\,000$.

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

Sample Input 1

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

Sample Output 1

```
0.0000000 0.0000000
1.0000000 0.0000000
-1.0000000 0.0000000
0.0000000 1.0000000
0.0000000 -1.0000000
```

Sample Input 2

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

Sample Output 2

```
40 40
40.7071067812 40.7071067812
41 40
41.7071067812 40.7071067812
40 41.4142135624
```

Problem D
Date Pickup

Richard and Janet are going on their first date. Richard has offered to meet her at home with his bicycle, and Janet tells him she will call when she is ready in 10 to 20 minutes. But Richard is an impatient person; while he could wait at home for Janet's signal, he might also leave early and travel around the neighbourhood for a bit, in order to minimise the time it takes him to reach her once she calls. Due to his impatience, once Richard is on his bicycle, he does not want to ride any slower than the legal speed limit, stop at intersections, or wait outside Janet's house (but he does not mind passing by Janet's house and returning to it later).

Given the directed graph representing the neighbourhood around Richard's and Janet's houses, Richard wants to devise a route around the neighbourhood (after an optional waiting period at his own house) which minimises the time that Janet has to wait in the worst case. He can travel for as long as he likes and visit each intersection as many times as he likes.

Janet will call Richard as soon as she is ready, and at that point Richard will take the shortest path to her that he can. Richard does not know exactly when Janet will be ready, but he knows it will be in somewhere between a and b minutes (not necessarily at a whole minute).

If Richard is passing through an intersection at the exact same instant Janet calls, the call is considered to happen before he chooses what to do at the intersection. For example, if he is passing by Janet's house at the moment she calls, he can immediately stop there and she does not have to wait for him at all.

It could happen that Janet never has to wait for w minutes, but that she might have to wait for $w - \epsilon$ minutes for arbitrarily small $\epsilon > 0$, if she calls Richard at some inopportune moment (say, nanoseconds after he has left an intersection). In this case, we still define the worst case waiting time to be w.

Input

The input consists of:

- One line with two integers a, b ($0 \leq a \leq b \leq 10^{12}$), indicating that Janet will be ready in at least a minutes and at most b minutes.
- One line with two integers n, m ($2 \leq n \leq m \leq 10^5$), the number of intersections and the number of roads in the neighbourhood. The intersections are numbered from 1 to n.
- m lines, each with three integers u, v and t ($1 \leq u, v \leq n$, $1 \leq t \leq 10^6$), indicating that there is a one-way road from intersection u to intersection v, and that it takes Richard exactly t minutes to travel along this road.

Richard's house is at intersection 1 and Janet's house is at intersection n. It is guaranteed that it is possible to travel from Richard's house to Janet's house, and that it is possible to exit each intersection through at least one road, even if that road just loops back to the same intersection.

Output

Output the time Janet has to wait in the worst case assuming she will be ready in at least a minutes and at most b minutes and Richard plans his route optimally.

It can be shown that the worst case waiting time is always an integer.

Sample Input 1

```
10 20
3 5
1 3 7
2 1 1
2 3 2
2 3 5
3 2 4
```

Sample Output 1

```
6
```

Sample Input 2

```
4 10
5 7
1 4 6
4 5 5
4 5 3
5 5 30
1 2 1
2 3 1
3 2 1
```

Sample Output 2

```
5
```

Problem E
Equality Control

In programming contest circles, one of the most important roles is that of the Chief Equality Officer (CEO). This person is responsible for making sure that every team has an equal chance of winning the contest. Since last year's NWERC the current CEO, Gregor, has thought at length about how to make the contest even more fair and equal.

His answer is to introduce a new programming language as the only one allowed for submissions. This way, no team will be disadvantaged by not having mastered any of the allowed languages. This language is called BALLOON, short for Building A Long List Of Ordinary Numbers. Its only data type is the list of integers. To keep the language fast, it contains only four instructions:

- `[`x_1`,...,`x_n`]` is the constructor for lists. It returns the integers inside the brackets in their given order.
- `concat(<Expr`$_1$`>,<Expr`$_2$`>)` returns a list of all the integers returned when evaluating the expression `<Expr`$_1$`>` followed by all of the integers returned when evaluating `<Expr`$_2$`>`.
- `shuffle(<Expr>)` returns a list of all the integers returned by `<Expr>`, rearranged according to a uniformly random permutation, i.e., each permutation of the elements is used with equal probability.
- `sorted(<Expr>)` returns a list of all the integers returned by `<Expr>`, rearranged into non-decreasing order.

As an example, consider the first expression of Sample Input 1. The two shuffle exressions both take the list `[1,2]` as input and return one of the lists `[1,2]` and `[2,1]`, each with probability 0.5 (independently of each other). The outer concat operator takes the two returned lists as its input and returns their concatenation. I.e., it returns one of the lists `[1,2,1,2]`, `[1,2,2,1]`, `[2,1,1,2]`, and `[2,1,2,1]`, each with probability 0.25.

Naturally, we cannot use byte-by-byte output comparison any more when teams submit their solutions in BALLOON, as its output is probabilistic. The judge server instead has to check whether a submitted program is equivalent to the sample solution created by the judges. Two programs are equivalent if for any list L of integers, both programs have an equal probability of returning L.

It is your task to determine whether two given BALLOON programs are equivalent.

Input

The input consists of:

- One line containing a string `A`, the first program.
- One line containing a string `B`, the second program.

Each program is a syntactically valid BALLOON program with between 3 and 10^6 characters, and contains neither spacing nor empty lists (i.e., the strings " " or "`[]`" do not occur in the input).

Each integer in each program is greater than 0 and less than 10^9.

Output

If the two programs are equivalent, output “`equal`”, otherwise output “`not equal`”.

Sample Input 1

```
concat(shuffle([1,2]),shuffle([1,2]))
shuffle([1,2,1,2])
```

Sample Output 1

```
not equal
```

Sample Input 2

```
sorted(concat([3,2,1],[4,5,6]))
[1,2,3,4,5,6]
```

Sample Output 2

```
equal
```

Sample Input 3

```
concat(sorted([4,3,2,1]),shuffle([1]))
concat(concat([1,2,3],shuffle([4])),sorted([1]))
```

Sample Output 3

```
equal
```

Problem F
Fastest Speedrun

The classic video game "Prince of Python" comprises n levels, numbered from 1 to n. You are going to speedrun this game by finishing all of the levels as fast as possible, and you can beat them in any order that you want.

You enter each level equipped with one of $n + 1$ magical items. In the beginning you only have item 0 in your inventory. Once you beat a level, you get to keep the item numbered the same as that level. For example, on finishing level 5, you obtain a mighty *Gauntlet of 5 Fingers* you may equip thereafter instead of the less-acclaimed *Sword of 0 Damage* you always start out with.

Beating a level can take different amounts of time depending on which item you take into the level with you. Higher-numbered items are more powerful, so if playing by the rules it is always at least as fast to finish the level with a higher-numbered item as with a lower-numbered item.

However, each level also has a shortcut left in by the developers. The shortcut for a level can be accessed by applying a specific item in an unconventional way. By doing so you can finish the level as fast as, or even faster than, if you had used any of the other items.

How long will it take you to beat all of the levels of the game?

Input

The input consists of:

- One line containing an integer n ($1 \leq n \leq 2\,500$), the number of levels.
- n lines, describing the levels.

  The ith such line starts with two integers x_i and s_i ($0 \leq x_i \leq n$, $1 \leq s_i \leq 10^9$), the shortcut item for level i and the completion time for level i when using the shortcut.

  The remainder of the line has $n + 1$ integers $a_{i,0}, \ldots, a_{i,n}$ ($10^9 \geq a_{i,0} \geq a_{i,1} \geq \ldots \geq a_{i,n} \geq s_i$), where $a_{i,j}$ is the completion time for level i when playing by the rules using item j.

Output

Output the minimum time it takes to beat, in any order, all of the levels in the game.

Sample Input 1

```
3
1 1 40 30 20 10
3 1 95 95 95 10
2 1 95 50 30 20
```

Sample Output 1

```
91
```

Sample Input 2

```
4
4 4 5 5 5 5 5
4 4 5 5 5 5 5
4 4 5 5 5 5 5
4 4 5 5 5 5 5
```

Sample Output 2

```
17
```

This page is intentionally left (almost) blank.

Problem G
Game Design

Carol enjoys playing with wooden games. The objective of the game that fascinates her the most is to tilt a maze, made out of 1 cm-by-1 cm blocks, in any of the four cardinal directions to move a ball to a hole in the centre at $(0, 0)$. As shown in Figure G.1, once the 1 cm wide ball starts moving, it keeps going until either it runs into a wooden block, or it falls into the hole—whichever comes first.

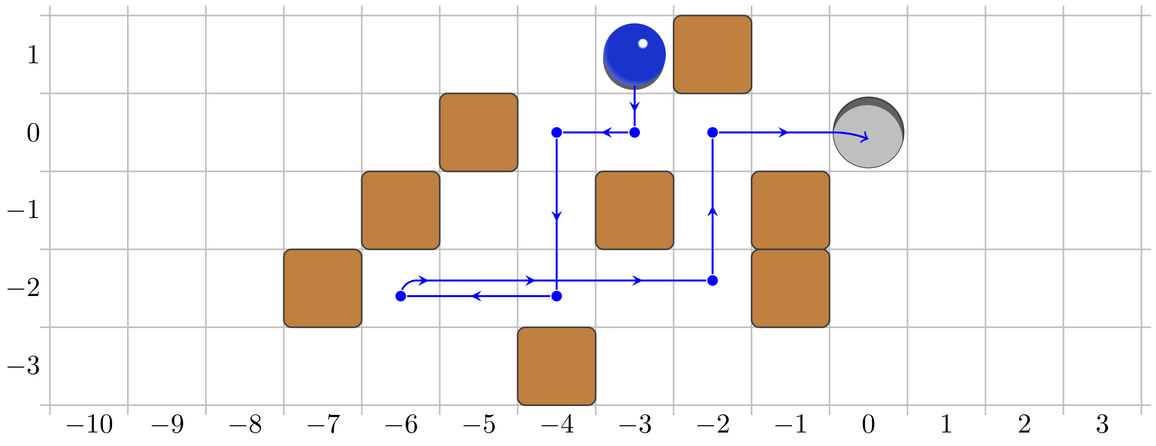


Figure G.1: Illustration of Sample Output 1.

Carol is interested in designing a maze of her own, and like any good game designer she already has a fixed solution in mind. This is given as a sequence of tilt moves which must all be followed in order. If any move causes nothing to happen, for example because the ball is up against a block in that direction or already in the hole, the solution does not count.

The ball can be placed anywhere to start. Carol will take care of adding a square border of blocks covering the rows and columns $10^9 + 1$ cells away from the centre in each direction.

Design a board which can be won by applying her sequence of moves.

Input

The input consists of:

- One line with a string s consisting of only the characters "`LRUD`" ($1 \leq |s| \leq 20$), the sequence of moves. These characters correspond to the directions $-x, +x, +y, -y$ respectively. No two consecutive characters in s are the same.

Output

If it is possible to create a maze with the given solution, first output x and y, the integer coordinates for the ball to start at. Then on the next line, output n, the number of blocks to use. On each of the remaining n lines, output s and t, the integer coordinates of a block.

Otherwise, output "`impossible`".

You may use at most $n \leq 10^4$ blocks. All coordinates used must be at most 10^9 in absolute value. No coordinate pair may be the same as the centre or any other coordinate pair. If there are multiple valid solutions, you may output any one of them.

Sample Input 1

```
DLDLRUR
```

Sample Output 1

```
-3 1
8
-1 -1
-1 -2
-2 1
-3 -1
-5 0
-6 -1
-7 -2
-4 -3
```

Sample Input 2

```
LRLRLRLRULD
```

Sample Output 2

```
1 1
5
2 1
2 0
-1 1
-1 0
-1 1000000000
```

Sample Input 3

```
LRLR
```

Sample Output 3

```
impossible
```

Problem H
Hard Drive

Pia is getting ready for her flight to the NWERC 2018 in Eindhoven. As she is packing her hard drive, she remembers the airline's ridiculous weight restrictions, which may pose a problem. You see, the hard drive is essentially a string of ones and zeros, and its weight depends on the number of "bit changes" in it: for any two adjacent bits storing two different values, the hard drive gets slightly heavier, so Pia cannot just store arbitrary information on it.

To make matters worse, the drive is so old that some bits are already broken and will always store zeros. The first bit will never be broken, but the last bit will always be.

Pia decides to treat this situation as a challenge: she is now trying to modify the information on the hard drive so that it has exactly the maximum number of bit changes permitted by the airline. However, the broken bits make this harder than expected, so she needs your help.

Find a bit pattern which can be stored on the hard drive and has exactly the desired number of bit changes.

Input

The input consists of:

- One line with three integers n, c, and b ($2 \leq n \leq 5 \cdot 10^5$, $1 \leq c, b \leq n - 1$), the size of the hard drive in bits, the desired amount of bit changes, and the number of broken bits. The positions on the hard drive are numbered from 1 to n.
- One line with b integers $z_1, \ldots, z_b$ ($2 \leq z_1 < z_2 < \ldots < z_b = n$), the positions of the broken bits on the hard drive.

Output

Output a bit string of length n, representing Pia's hard drive and containing exactly c bit changes. If there are multiple valid solutions, you may output any one of them. It is guaranteed that at least one solution exists.

Sample Input 1	Sample Output 1
5 2 3 2 3 5	00010

Sample Input 2	Sample Output 2
7 4 2 2 7	0010110

This page is intentionally left (almost) blank.

Problem I
Inflation

For NWERC 2018, the organisers have done something rather special with the balloons. Instead of buying balloons of equal size, they bought one balloon of every integer size from 1 up to n. A balloon of size s has a capacity of s decilitres.

To avoid inflating the balloons by hand, the organisers also bought n helium gas canisters. Each canister can only be used to inflate one balloon, and must be emptied completely into that balloon (it is not possible to disconnect a canister from a balloon before the canister has been fully used).

Unfortunately the gas canisters were bought at a garage sale, and may contain differing amounts of helium. Some may even be empty! To make the best of this challenging situation, the canisters will have to be paired with the balloons smartly.

The organisers want to assign all of the gas canisters to separate balloons, such that the balloon that is inflated the least (relative to its capacity) still contains the maximum possible fraction of helium inside. What is the maximum such (minimum) fraction that is possible?

Balloons filled beyond their capacity will explode. Explosions are upsetting and must be avoided.

Input

The input consists of:

- One line with the integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of balloons and gas canisters.
- One line with n integers $c_1, \ldots, c_n$ ($0 \leq c_i \leq n$ for each i), the amounts of helium in the gas canisters, in decilitres.

Output

If it is possible to fill all the balloons without any exploding, output the maximum fraction f such that every balloon can be filled to at least f of its capacity. Otherwise, output "`impossible`".

Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
6 6 1 3 2 2 3	0.6

Sample Input 2	Sample Output 2
2 2 2	impossible

Sample Input 3	Sample Output 3
5 4 0 2 1 2	0

This page is intentionally left (almost) blank.

Problem J
Jinxed Betting

Julia is betting on a large sporting competition involving matches between pairs of teams. There are no parallel matches and each bettor receives one point for every correct bet they make. Julia had a good streak and is in the lead. Now she worries that her good luck may be turning, and decides to change her strategy.

She collaborates with a betting shop owner who tells her the bets made by everyone else. Whenever Julia makes a bet, she first checks the bets of all bettors with the most points so far (except herself of course) and then chooses the same team as the majority. In the case of a tie, she bets on her favourite of the two teams in the game.

Julia wants to know for how many more matches she is guaranteed to stay in the lead in the worst case (i.e., no matter what bets the others make or what the outcomes of the games are). For this problem we consider Julia to be in the lead if there is no other bettor that has strictly more points than her.

For example, suppose as in Sample Input 1 that Julia initially has three points, and there are two other bettors with three and two points respectively. For the first match, she will make the same bet as the other bettor with three points. If this person bets on the losing team and the other bets on the winning team all players have an equal score of three points. If for the next match the other two persons bet on different teams, Julia places the bet on her favourite, which of course may lose. Thus after two more matches she might lose the lead.

Input

The input consists of:

- One line with an integer n ($3 \le n \le 10^5$), the number of people who place their bets.
- One line with n integers $p_1, \ldots, p_n$ ($0 \le p_i \le 10^{16}$ for each i), the points of all people who play the betting game. The first of these numbers corresponds to the score of Julia. You may assume that no other score exceeds Julia's score in the beginning.

Output

Output the number of matches for which Julia is guaranteed to stay in the lead.

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

Sample Input 2
```
5
8 4 3 5 2
```
Sample Output 2
```
6
```

This page is intentionally left (almost) blank.

Problem K
Kleptography

John likes simple ciphers. He had been using the "Caesar" cipher to encrypt his diary until recently, when he learned a hard lesson about its strength by catching his sister Mary browsing through the diary without any problems.

Rapidly searching for an alternative, John found a solution: the famous "Autokey" cipher. He uses a version that takes the 26 lower-case letters 'a'–'z' and internally translates them in alphabetical order to the numbers 0 to 25.

The encryption key k begins with a secret prefix of n letters. Each of the remaining letters of the key is copied from the letters of the plaintext a, so that $k_{n+i} = a_i$ for $i \geq 1$. Encryption of the plaintext a to the ciphertext b follows the formula $b_i = a_i + k_i \bmod 26$.

Mary is not easily discouraged. She was able to get a peek at the last n letters John typed into his diary on the family computer before he noticed her, quickly encrypted the text document with a click, and left. This could be her chance.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 30$, $n + 1 \leq m \leq 100$), where n is the length of the keyword as well as the number of letters Mary saw, and m is the length of the text.
- One line with n lower-case letters, the last n letters of the plaintext.
- One line with m lower-case letters, the whole ciphertext.

Output

Output the plaintext of John's diary.

Sample Input 1

```
5 16
again
pirpumsemoystoal
```

Sample Output 1

```
marywasnosyagain
```

Sample Input 2

```
1 12
d
fzvfkdocukfu
```

Sample Output 2

```
shortkeyword
```

This page is intentionally left (almost) blank.