SOUTH PACIFIC ALGORITHMIC ROUNDS (SPAR)

ROUND 1 MARCH 2, 2024

Contest Problems

Problem set contains 14 problems over 30 pages

This page is intentionally left (almost) blank.

South Pacific Competitive Programming Association

wisetech global

Problem A

This is SPAR-ta: The 300 Minutes

Time limit: 1 second

Welcome to SPAR! The South Pacific Algorithmic Rounds are a chance for students from the South Pacific to practice problem solving and programming contests. Teams of students (normally three) have 5 hours (300 minutes) to complete as many problems as possible. Each problem has some descriptive text (like this text) describing the problem, and often doing this in obfuscated way to make you think carefully about what the problem actually is. To clarify what the problem is asking, some sample inputs are given, with matching outputs. The inputs and outputs are always text, and the inputs are read from the standard input stream (so, for example, in python `input()` will evaluate as each successive line of the input). The output is always written as standard output (so, for example, in python the line `print(ans)` will print the value of `ans`), and the system will check to see that it matches the expected answer).

Make sure that the format is exactly right, because if you print the wrong number of decimal places, forget to capitalise a word, or leave out a full stop, the system will judge your answer to be wrong. When you submit a problem, the system will compile your code and then run it against a number of test cases. For each test case you can get the right answer (**Acc**epted), produce an incorrect answer (**W**rong **A**nswer)), have a runtime error (**R**un **E**rror), or exceed the time limit (**T**ime **L**imit **E**xceeded). Each problem has a time limit within which each test case must be solved. The time limit for this problem is one second. You will not be given any information other than you passed all test cases (Acc), or at least one test case did not pass. If a test case did not pass, you are told whether it was a wrong answer, run time error or a time limit exceeded.

There are a number of problems to solve: this round there are 14. Some are very easy, and some are very difficult, but each is worth a single point, and who ever solves the most problems wins. Because draws are very common, when two teams have the same score, the winning team is the team with the fewest penalty minutes. Every time you solve a problem you have the time (in minutes) since the start of the contest added to your penalty, plus 20 minutes for every time you unsuccessfully attempted that problem.

For this problem, you should create a program to work out your maximum score for a problem set. We assume that you have very good coding, reading and estimation skills, so that you have read the problem set, you know the answers to all the problems, and you only ever submit correct answers; all that's left to do is to type them up! Your elite estimation skills mean that you known exactly how long it will take you to type in each solution, so now you need to work out the best score you can get in 300 minutes. Your score should have the highest number of problems solved possible, and the corresponding lowest penalty that you can get for that score. That is, the sum of the times should be less than or equal to 300; the score is the number of problems solved, and the penalty is the sum of the completion times.

Input

The input will consists of two lines: the first has a single integer, n, where $1 \leq n \leq 50$, being the number of problems in the set; the second line then has a list of n integers, $p_1 p_2 \ldots p_n$ where $1 \leq p_i \leq 300$. The number p_i is the number of minutes required to solve the i^{th} problem in the set.

Output

Print a pair of numbers $N P$ on a single line, separated by a space, where N is the number of problems solve, and P is the penalty.

Sample Input 1	Sample Output 1
3 100 200 300	2 400

Sample Input 2	Sample Output 2
5 20 20 50 120 100	4 340

Problem B
Bidirectional Code
Time limit: 1 second

For communication through space between earth and satellites, one cannot simply transmit a message in the same way as cellular communication like 4G. Because of the extremely long distance a signal travels, the message might be distorted by noise.

Throughout the years, researchers have found ways to bypass this problem, and the solution lies in introducing redundant data. This gives the receiver a method to test if the received data contains an error, in which case it can ask the sender to transmit the message again, or it might be able to recover the original message if there was only a small error. This area of research is called Coding Theory.

We created a new redundant system to prevent mistakes. Now, to send a number, you simply find a way to express it as a sum of palindromic numbers, and send each palindromic number as you would normally do. The receiver may now check if it received a number which was not palindromic, in which case there was an error.

To keep the communication efficient enough, the institute has added the constraint that a number can only be broken down into the sum of at most 10 palindromic numbers. You must find a way to break any number down into palindromic numbers.[1].

Input

- One line containing a single integer n ($1 \leq n < 10^{18}$), the number you want to write as a sum of palindromic numbers.

Output

First, output a line with a single number $1 \leq k \leq 10$, the number of palindromic numbers you need. Then follow k lines each containing a palindromic number.

Sample Input 1
```
1100000
```

Sample Output 1
```
2
645546
454454
```

Sample Input 2
```
1000
```

Sample Output 2
```
5
1
99
1
898
1
```

[1]A recent paper has shown that every positive integer is a sum of three palindromic numbers.

This page is intentionally left (almost) blank.

Problem C
Corrupt Judge
Time limit: 1 second

You are organising a programming competition in which the rank of a team is first determined by how many problems they have solved. In case of a tie, the team with the lowest time penalty is ranked above the other. However, contrary to the UKIEPC, the time penalty is equal to t if the *latest* accepted submission was submitted in the tth minute, or 0 if no problem was solved.

For example, if team A solved their first problem in the 5th minute, their second problem in the 10th minute and their third problem in the 60th minute, then their time penalty is 60. If team B also solved three problems, in the 30th, 40th and 50th minute, their time penalty is 50 and they would rank above team A.

The contest has finished and you would like to enter the final standings. However, due to a corrupted file you have lost part of the scoreboard. In particular, the column indicating how many problems each team has solved is gone. You do still have the time penalties of all the teams and know that they are in the right order. You also remember how many problems the contest had. You wonder whether, given this information, it is possible to uniquely reconstruct the number of problems that each team has solved.

Input

- One line containing two integers: n ($1 \leq n \leq 10^4$), the number of teams participating, and p ($1 \leq p \leq 10^4$), the number of contest problems.
- n lines with on line i the time score t_i in minutes ($0 \leq t_i \leq 10^6$) of the team that is ranked in the ith place.

A positive time score of t indicates that a team has submitted their last accepted submission in the tth minute. A time score of 0 indicates that a team hasn't solved any problem.

The input always originates from a valid scoreboard.

Output

If it is possible to uniquely reconstruct the scores of all the teams, output n lines containing the number of problems that the ith team has solved on the ith line. Otherwise, output "`ambiguous`".

Sample Input 1	Sample Output 1

```
9 3
140
75
101
120
30
70
200
0
0
```
	```
	3
	2
	2
	2
	1
	1
	1
	0
	0
	```

Sample Input 2

```
6 3
100
40
40
50
0
0
```

Sample Output 2

```
ambiguous
```

Problem D
Divvying Up
Time limit: 2 seconds

A solid competitive programming team can rack up a lot of prize money. Knowing how strong your team is, you are certain to win a lot of contests, so you had better sit down now and check that everybody will receive the same fair distribution of winnings.

You will participate in multiple contests, and at the end of each one receive a set amount of prize money. You can distribute any amount to each of the three members of your team each time, but by the end everyone must have the same amount of total winnings.

Can you distribute the winnings such that everyone gets an equal amount by the end?

Input

- One line containing the number of contests, n $(1 \leq n \leq 10^4)$.
- One line containing the prize purse for each contest, $w_1 \ldots w_n$ $(1 \leq w \leq 10^5)$.

Output

Output `yes` if the winnings can be distributed equally between three contestants, otherwise `no`.

Sample Input 1

```
2
10 3
```

Sample Output 1

```
no
```

Sample Input 2

```
3
9 8 7
```

Sample Output 2

```
yes
```

This page is intentionally left (almost) blank.

South Pacific Competitive Programming Association

wisetech global

Problem E
Elevator Pitch
Time limit: 2 seconds

You are in charge of ensuring all building designs meet accessibility requirements. As law dictates, every part of your building should be reachable for wheelchair users, which means elevators will have to be installed. You are given the blueprints of the company's current project and have to determine the minimum number of elevators required.

The floor plan is laid out on a square grid and the blueprints tell you the number of floors above any given square. You can place an elevator at any square, which stops at all floors of that square. A wheelchair user can move up and down between floors using the elevators and can freely move to any of the four adjacent squares on the same floor. Buildings do not connect diagonally.

The image below shows the second sample input. Designs can consist of multiple buildings; this one contains three buildings. The design requires two elevators: one for the pyramid-shaped building and one for the tall tower. The small building of height one does not require an elevator, since it only has a ground floor.

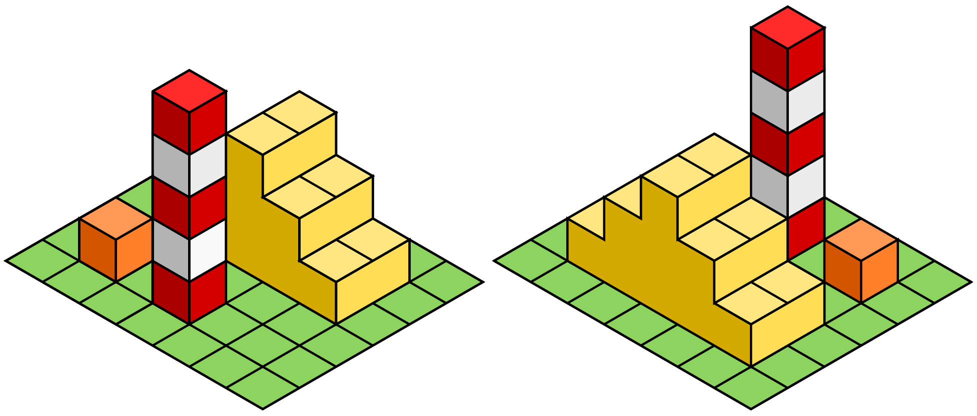
Figure E.1: A visualisation of the second sample input.

Input

- One line containing integers h and w ($1 \leq h, w \leq 500$), the height and width of the grid.
- h lines of w integers each, where $x_{i,j}$ ($0 \leq x_{i,j} \leq 10^9$), the jth integer on the ith line, denotes the number of floors at position (i, j) of the grid.

Output

Output the minimum number of elevators you need to build to be able to reach every part of the building(s) in the grid.

Sample Input 1	Sample Output 1

3 3
1 2 3
0 0 4
7 6 5
	1

Sample Input 2

```
6 7
0 0 0 0 0 0 0
0 1 2 3 2 1 0
0 1 2 3 2 1 0
0 0 0 0 0 0 0
0 1 0 5 0 0 0
0 0 0 0 0 0 0
```

Sample Output 2

```
2
```

Sample Input 3

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

Sample Output 3

```
4
```

Problem F
Family Fares
Time limit: 3 seconds

After a long time apart, your family will gather next year for a reunion in an idyllic village in the centre of the country. Since everybody lives apart, most will need to travel by train.

You are in charge of finding the best deal on tickets. Everyone must take an optimal route, that is to say they may only travel a route if no other route is shorter.

Two types of ticket are available: *individual* or *group*. All tickets come with a start and destination between which to travel. Individual tickets are unlimited and the price is equal to the shortest distance in kilometres between stations.

Group tickets are more complicated. First, you may only buy at most one and it must be for a set list of people. There is no limit to the number of people named, but all must be present. The ticket is priced according to the number of named persons.

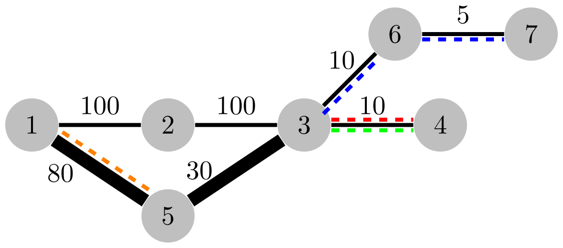

Figure F.1: Sample 2. Group or individual tickets are shown by thick or thin lines, respectively.

Input

- One line with four integers: n ($2 \leq n \leq 1000$), the number of stations, m ($n-1 \leq m \leq 10^5$), the number of connections between stations, p ($1 \leq p \leq 100$), the number of family members, and g ($1 \leq g \leq 10^6$), the cost per person of a group ticket.
- One line with p integers v_i ($1 \leq v \leq n$), meaning that family member i starts at station v_i.
- m further lines, each with three integers a, b, and c ($1 \leq a, b \leq n$, $a \neq b$, and $1 \leq c \leq 10^6$), indicating that there is a bidirectional connection between stations a and b with a length of c kilometres.

Each pair of distinct stations has at most one direct connection and every station can be reached from any other station. Station number 1 serves the idyllic village.

Output

Output the total amount you must spend so that every family member can travel from their starting station to the idyllic village.

Sample Input 1

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

Sample Output 1

```
35
```

Sample Input 2

```
7 7 4 10
5 4 4 7
1 2 100
2 3 100
3 4 10
1 5 80
3 5 30
3 6 10
6 7 5
```

Sample Output 2

```
145
```

Sample Input 3

```
4 5 2 10
2 4
1 2 20
2 4 5
1 3 20
3 4 5
1 4 30
```

Sample Output 3

```
25
```

Problem G

Generators

Time limit: 2 seconds

The volcanic island of Fleeland has never had a proper electric net, but finally the administration of the island have agreed to build the island's power plants and network.

On the island's coast are its n cities. The administration has surveyed the cities and proposed m of them as possible locations for a power plant, with the ith proposal stating that the company can build a plant in city c_i for cost a_i.

These power plants are very modern and a single plant could power the whole island, but the volcano makes building power lines across the island a dangerous affair. For $1 \leq i < n$, the company can build power lines between cities i and $i + 1$ for a cost of b_i, and between cities n and 1 for a cost of b_n. A city will receive power if it contains a power plant or is connected to a city with a power plant via power lines.

What is the cheapest way to power all the cities on the island?

Input

- One line containing two integers n ($3 \leq n \leq 10^5$) and m ($1 \leq m \leq n$), the number of cities and the number of possible locations for a power plant.
- Then follow m lines, the ith of which contains c_i ($1 \leq c_i \leq n$) and a_i ($1 \leq a_i \leq 10^9$), the ith possible location for a power plant, and the cost to build it.
- Then follows a line containing n integers b_i ($1 \leq b_i \leq 10^9$), the costs of building the power lines.

The values of $c_{1,\ldots,n}$ are unique and given in strictly increasing order.

Output

Output the minimal cost of powering all cities on the island.

Sample Input 1

```
3 2
1 100
2 200
150 300 150
```

Sample Output 1

```
400
```

Sample Input 2

```
3 2
1 100
2 200
300 300 150
```

Sample Output 2

```
450
```

This page is intentionally left (almost) blank.

Problem H
Haughty Cuisine
Time limit: 1 second

As a waiter, your favourite question from an indecisive punter is “I’m not sure, what would you recommend?” — so much so, in fact, that you decided to automate away the answer to avoid having to spend any brain cycles on this question ever again.

You have the list of all set menus for today and you are going to simply pick one at random. As long as your recommendation corresponds to a list of items on a set menu, everything will be fine.

Input

- One line containing a single integer $1 \leq n \leq 100$, the number of set menus.
- n lines, one for each menu. Each of these lines contains a single integer $1 \leq d \leq 42$, followed by a list of d dishes that the meal consists of.

Each dish is described using at most 20 lowercase Latin characters.

Output

Output one line containing m, the number of dishes that you recommend, followed by m lines containing the dishes you recommend.

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

Sample Input 1

```
3
2 bigburger fries
2 pizza garlicbread
2 macaroni cheese
```

Sample Output 1

```
2
garlicbread
pizza
```

Sample Input 2

```
4
2 pasta pizza
3 icecream sweets pasta
1 megapizza
2 icecream pizza
```

Sample Output 2

```
3
pasta
icecream
sweets
```

This page is intentionally left (almost) blank.

Problem I
Incomplete Sort
Time limit: 2 seconds

Merge sort is a sorting algorithm. It works by splitting an array in half, sorting both halves recursively and then merging those halves together to sort the entire array. Your friend is working on an implementation of the merge sort algorithm, but unfortunately he is not quite there yet: he can only sort half of the array! In great despair he turns to you for help: can you use his unfinished code to write an algorithm that sorts an array completely?

In its current state, your friend's code is a sorting function that can be run on arbitrary sub-arrays, as long as it is precisely half as long as the original array. It then correctly sorts this sub-array. You decide to play around with this function, so you start with a jumbled array and try to sort it (see figure). After choosing 3 sub-arrays and using them as input for the sorting function, you end up with a sorted array. Interestingly, it seems that no matter what the original array you use is, you can always sort it completely by invoking your friend's sorting function only 3 times. You decide that this makes for a good challenge: you want to extend the code to work for a full array, making at most three calls to the sorting function.

Now you need to figure out which sub-arrays to sort! Given an array of length n, output at most three sub-arrays of length $\frac{1}{2}n$ so that sorting these sub-arrays in order will result in a sorted array. It is guaranteed that this is always possible.

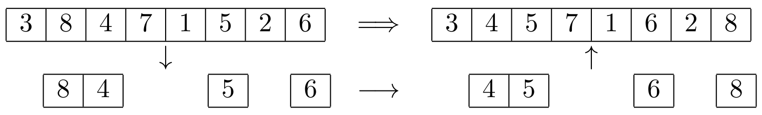

Figure I.1: First sorting step of sample output 1

Input

- One line containing a single integer n ($4 \leq n \leq 10^5$) divisible by 4, the length of the array.
- One line containing n unique integers a ($1 \leq a \leq n$), the array to be sorted.

Output

The output consists of:

- One line containing the number of function calls f ($0 \leq f \leq 3$).
- f lines, each containing $\frac{1}{2}n$ unique integers i ($1 \leq i \leq n$), the indices determining the sub-array to be sorted at each of the function calls.

If there are multiple valid solutions, you may output any one of them. You do not have to minimise f.

Sample Input 1

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

Sample Output 1

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

Sample Input 2

```
4
1 4 3 2
```

Sample Output 2

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

Sample Input 3

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

Sample Output 3

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

Problem J
Jigsaw
Time limit: 2 seconds

You have found an old jigsaw puzzle in the attic of your house, left behind by the previous occupants. Because you like puzzles, you decide to put this one together. But before you start, you want to know whether this puzzle was left behind for a reason. Maybe it is incomplete? Maybe the box contains pieces from multiple puzzles?

If it looks like a complete puzzle, you also need to know how big your work surface needs to be. Nothing worse than having to start a jigsaw over because you started on a small table.

The box does not tell you the dimensions $w \times h$ of the puzzle, but you can quickly count the three types of pieces in the box:

- Corner pieces, which touch two of the edges of the puzzle.
- Edge pieces, which touch one of the edges of the puzzle.
- Centre pieces, which touch none of the edges of the puzzle.

Do these pieces add up to a complete jigsaw puzzle? If so, what was the original size of the jigsaw puzzle?

Input

- One line containing three integers c, e, and m ($0 \leq c, e, m \leq 10^9$), the number of corner pieces, edge pieces, and centre pieces respectively.

Output

If there exist numbers w and h satisfying $w \geq h \geq 2$ such that the original size of the jigsaw puzzle could have been $w \times h$, then output a single line containing w and h. Otherwise, output "`impossible`".

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

Sample Input 1	Sample Output 1
`4 8 4`	`4 4`

Sample Input 2	Sample Output 2
`4 10 14`	`impossible`

Sample Input 3	Sample Output 3
`4 12 6`	`impossible`

Sample Input 4	Sample Output 4
`4 2048 195063`	`773 255`

Problem K
Kleptocrat
Time limit: 5 seconds

Your company has a policy that every employee should be refunded an amount of money proportional to the shortest distance between their home and office. This causes the loophole that many employees intentionally move very far away to claim the maximum possible reimbursement.

One employee has taken this policy way too far and is in danger of bankrupting you. You must find a way to stop this before cancelling the policy next year. However, the rules are strict: as long as the employee keeps track of the distances they have travelled, you are forced to reimburse them.

Suddenly you have a flash of inspiration: nowhere does it say that you have to use the *Euclidean* distances! You start working on more subtle distance functions and now you have a first prototype: XOR distance. The length of a path is defined as the XOR of the lengths of the edges on the path (as opposed to the sum). The distance between two locations is defined as the length of the shortest path between them.

You will need to test this principle on the transport network with the locations of each of your employees in turn.

Input

- One line containing three integers n ($2 \leq n \leq 10^4$), m ($n-1 \leq m \leq 10^5$), and q ($1 \leq q \leq 10^5$), the number of nodes, edges, and questions respectively.
- m lines describing an edge. Each line consists of three integers x, y, w ($1 \leq x, y \leq n$, $x \neq y$ and $0 \leq w \leq 10^{18}$), indicating that there is an undirected edge of length w between nodes x and y.
- q lines describing a question. Each line consists of two integers a, b ($1 \leq a, b \leq n$) asking for the shortest distance between nodes a and b.

Between every pair of distinct nodes, there is at most one edge, and every node can be reached from any other node.

Output

For every question, output one line containing the shortest distance between nodes a and b.

Sample Input 1

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

Sample Output 1

```
1
1
0
```

Sample Input 2

```
7 10 5
1 2 45
2 3 11
2 4 46
3 4 28
3 5 59
3 6 12
3 7 3
4 5 11
5 6 23
6 7 20
1 4
2 6
3 5
1 7
5 5
```

Sample Output 2

```
1
5
0
5
0
```

Problem L
Lost Map
Time limit: 20 seconds

An amateur Viking historian needs your help finding the silver left by Egill Skallagrímsson, of Egil's saga. She has found two old treasure maps that are supposed to lead to it. A treasure map is a list of instructions of the form "*direction* k", where *direction* can be "n", "s", "e", or "w". The maps are sadly old, so some of the instructions are missing and we represent them with a simple "?" instead.

The first map is larger while the second map is a smaller fragment. She wants to know how she can overlay her maps such that they coincide.

Two maps coincide if the corresponding instructions are either identical or at least one of them is lost to time. All instructions must have a corresponding instruction on the other map when overlaying the maps.

Input

- The first line of the input contains two integers, $1 \leq m < n \leq 4 \cdot 10^5$.
- The next n lines describe the first map with each containing either "?", or "(n—s—e—w)" followed by the number of steps s $(1 \leq s \leq 7)$.
- The next m lines describe the second map with each containing either "?", or "(n—s—e—w)" followed by the number of steps s $(1 \leq s \leq 7)$.

Output

Output the number of indices such that if the second map was overlaid at this index on the first map then they would coincide.

Sample Input 1

```
4 3
n 4
e 1
?
s 5
?
e 1
?
```

Sample Output 1

```
2
```

Sample Input 2

```
4 3
n 4
e 1
w 3
s 5
?
e 1
?
```

Sample Output 2

```
1
```

This page is intentionally left (almost) blank.

Problem M
Moderate Pace
Time limit: 2 seconds

An ultra-marathon is a race that takes place over an uncomfortably long distance and time, typically lasting for five hours or more. You are part of a group of three ultra-marathon runners looking to place in this year's Great South-to-North run from Plymouth to Aberdeen.

You have a set number of days until the next race to train. You will all train together, as training alone can be dangerous. As everyone has their own schedule in mind for how many kilometres to run per day, this will not be easy, you will have to compromise.

The fairest option is to look at each day individually, examine the three options for how far to run, and to take the median one. That is to say, the option taken for each day should be one that is not be greater or lesser than both of the other possibilities at the same time.

Input

- A line with the integer n ($1 \leq n \leq 1000$), the number of days of training.
- A line with n integers $k_{1,\ldots,n}$ ($0 \leq k \leq 10^6$), your ideal daily distances.
- A line with n integers $a_{1,\ldots,n}$ ($0 \leq a \leq 10^6$), your first colleague's ideal daily distances.
- A line with n integers $b_{1,\ldots,n}$ ($0 \leq b \leq 10^6$), your second colleague's ideal daily distances.

Output

Output a plan for the n days as n integers, where the distance for every day corresponds to the median of choices for that day.

Sample Input 1
```
4
1 2 3 4
4 3 2 1
2 2 2 2
```

Sample Output 1
```
2 2 2 2
```

Sample Input 2
```
6
3 1 4 1 5 9
2 7 1 8 2 8
1 6 1 8 0 3
```

Sample Output 2
```
2 6 1 8 2 8
```

This page is intentionally left (almost) blank.

Problem N
Nibbling Piranhas
Time limit: 2 seconds

The aquarium at which you work is hoping to expand its meagre selection of aquatic life, but lacks the funds to do so. You have been tasked to help promote the aquarium by taking photos of the two exhibits. Taking the first photo went swimmingly, because the catfish were very cooperative. For the piranhas, you have an arrangement of piranhas in mind that will look great on the photo. However, the only way to get the piranhas to move is by recklessly sticking your finger into the water to lure the piranhas. Your goal is to move the piranhas to the desired positions as quickly as possible without losing your finger in the process.

The piranha exhibit can be divided into positions $1, \ldots, n$ from left to right. The exhibit contains k piranhas and every position is occupied by at most one piranha. You can stick your finger into any unoccupied position. This will lure the nearest piranha to the left of your finger and the nearest piranha to the right of your finger. These piranhas will swim towards your finger, moving forward one position per second. All other piranhas simply stay in place. A piranha will bite your finger if it reaches the same position, so you must pull your finger away before this happens. Pulling your finger away and sticking it into a different position does not take any time.

For example, suppose there are piranhas at positions 2, 7 and 9. If you stick your finger into the water at position 4, the piranhas will be at positions 3, 6 and 9 after one second. You now have to pull your finger away to prevent the piranha at position 3 from biting your finger one second later. If you now stick your finger into the water at position 1, only the piranha at position 3 will move and will end up at position 2 after one second.

Input

- One line containing two integers n ($1 \leq n \leq 1000$), the number of positions, and k ($1 \leq k \leq n$), the number of piranhas.
- One line containing k integers $1 \leq p_1 < \ldots < p_k \leq n$, the current positions of the piranhas.
- One line containing k integers $1 \leq d_1 < \ldots < d_k \leq n$, the desired positions of the piranhas.

Output

Output the minimum number of seconds needed to get all of the piranhas at the desired positions. If it is impossible to do so, output "impossible".

Sample Input 1
```
9 3
3 7 9
3 5 9
```
Sample Output 1
```
4
```

Sample Input 2
```
8 3
1 5 8
2 4 7
```
Sample Output 2
```
impossible
```

Sample Input 3

```
20 6
1 4 7 10 13 20
2 5 8 11 14 17
```

Sample Output 3

```
17
```