2019 ECNA Regional Contest

ECNA 2019

October 26, 2019

ICPC North America Regionals 2019

icpc international collegiate programming contest

icpc global programming tools sponsor

icpc north america sponsor

ICPC East Central NA Regional Contest

Problems

A Retribution!
B Bio Trip
C Cheese, If You Please
D Follow the Bouncing Ball
E Just Passing Through
F Musical Chairs
G Out of Sorts
H Remainder Reminder
I Square Rooms
J Taxed Editor
K Where Have You Bin?

Do not open before the contest has started.

International Collegiate Programming Contest
2019 East Central Regional Contest
Grand Valley State University
University of Cincinnati
University of Windsor
Youngstown State University
October 26, 2019

Rules:

1. There are **eleven** problems to be completed in **5 hours**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. When displaying results, follow the format in the Sample Output for each problem. Unless otherwise stated, all whitespace in the Sample Output consists of exactly one blank character.
4. The allowed programming languages are C, C++, Java, Python 2 and Python 3.
5. All programs will be re-compiled prior to testing with the judges' data.
6. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation).
7. Programs will be run against multiple input files, each file containing a single test case.
8. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
9. All communication with the judges will be handled by the Kattis environment.
10. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A
Retribution!

The coaches in a certain regional are fed up with the judges. During the last contest over 90% of the teams failed to solve a single problem—in fact, even half the judges found the problems too hard to solve. So the coaches have decided to tar and feather the judges. They know the locations of all the judges as well as the locations of tar repositories and feather storehouses. They would like to assign one repository and one storehouse to each judge so as to minimize the total distances involved. But this is a hard problem and the coaches don't have time to solve it (the judges are evil but not stupid—they have a sense of the unrest they've fomented and are getting ready to leave town). So instead they've decided to use a greedy solution. They'll look for the smallest distance between any tar repository and any judge location and assign that repository to that judge. Then they'll repeat the process with the remaining repositories and judges until all the judges have a repository assigned to them. After they're finished with the tar assignments they'll do the same with the feather storehouses and the judges. Your job is to determine the total distances between repositories and storehouses and their assigned judges.

All judges, tar repositories and feather storehouses are numbered $1, 2, \ldots$. In case of any ties, always assign a repository/storehouse to the lowest numbered judge first. If there is still a tie, use the lowest numbered repository/storehouse.

Better hurry up—an unmarked van has just been spotted pulling up behind the judges' room.

Input

Input starts with a line containing three positive integers: $n\ m\ p$ ($1 \leq n \leq m, p \leq 1\,000$), representing the number of judges, tar repositories and feather storehouses, respectively. Following this are n lines, each containing two integers $x\ y$ ($|x|, |y| \leq 10\,000$) specifying the locations of the n judges, starting with judge 1. This is followed by m similar lines specifying the locations of the tar repositories (starting with repository 1) and p lines specifying the locations of the feather storehouses (starting with storehouse 1).

Output

Output the the sum of all distances between judges and their assigned tar repositories and feather storehouses, using the greedy method described above. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
2 2 2 1 0 2 0 0 0 3 0 1 1 2 1	4.0

This page is intentionally left blank.

Problem B
Bio Trip

Ollie MacDonald is in charge of a biological reserve used to study the nesting habits of birds. Scattered throughout the reserve are various types of nesting boxes, and one of Ollie's tasks is to regularly check on the boxes. The only way to travel between the boxes is via a set of dirt roads which meet at various junctions in the reserve, and the only way for Ollie to travel on these roads is with an old, beat up tractor (funding hasn't been the greatest for the past few years). Recently, a problem has developed with the steering mechanism of the tractor which limits the turning angles available. Since the junctions are relatively small, the roads that Ollie can take when entering a junction can be restricted. For example, in Figure B.1, if Ollie enters the junction from road A he may be able to leave on either roads B or C, but if he enters via road B, the constrained turning angle may only allow him to leave on road A (the same could happen if he entered on road C).

In addition, due to uneven terrain, it is possible that traveling a road in one direction may take longer than traveling the road in the opposite direction.

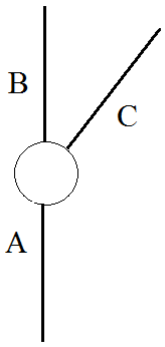

Figure B.1: Sample junction.

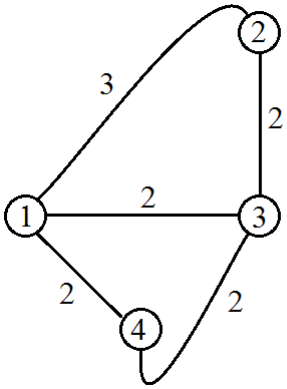

Figure B.2: Map of Sample Input 1.

As an example, Figure B.2 illustrates the map of the reserve described in the first sample input. In this map, there are two ways to reach junction 3 and return to the biostation: visit all junctions in order in 9 minutes, or travel to junction 3 first, turn left and travel to junction 2 and then back to the start, for a total of 7 minutes.

Note that in this map, the travel times between two junctions is the same no matter which direction Ollie travels. This is not always the case with other maps.

Note also that the tractor cannot travel from the start to junction 3 and then turn right and head for junction 4, since that would require a 135-degree turn but the tractor can only make a 90-degree turn. Ollie also cannot make a U-turn — a 180-degree turn — and return to the biostation directly.

The junction at the biostation is large enough for Ollie to turn the tractor in any direction, so he can take any road leading from the biostation.

Given a layout of the roads, turning restrictions on the tractor and a destination box which Ollie needs to visit, he would like to know the minimum time he needs in order to get to the box and back.

Input

Input starts with a line containing four integers n d α_1 α_2, where n $(2 \leq n \leq 1\,000)$ is the number of junctions (numbered 1 to n), d is the junction containing the bird box to visit, and α_1 and α_2 $(0 < \alpha_1, \alpha_2 \leq 180)$ specify the allowed turning angles in degrees (see Figure B.3). The biostation is at junction 1 and is where Ollie's journey both starts and ends. Following this are n lines specifying the dirt roads. Each of these lines has the form $m\ d_1\ t_1\ a_1\ d_2\ t_2\ a_2 \ldots d_m\ t_m\ a_m$. The i^{th} of these lines indicates that there are m dirt roads intersecting at junction i. The first of these roads ends at junction d_1, needs t_1 minutes to travel and leaves junction i at angle a_1 (where 0 is east, 90 is north, etc.); the second of these roads ends at junction d_2, needs t_2 minutes to travel and leaves junction i at angle a_2, etc. The maximum value of m for any junction is 5 and the maximum value for any t_i is 20.

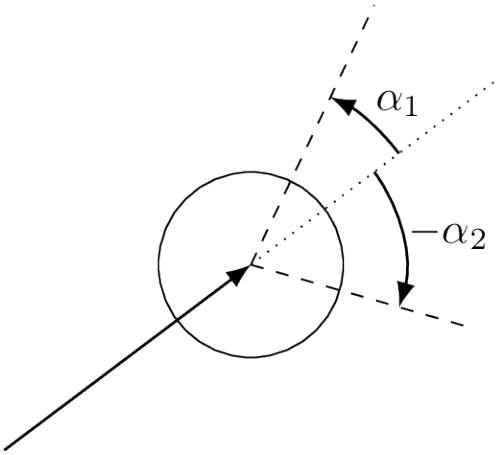

Figure B.3: Turning angle specification.

Output

Output the minimum time for Ollie to travel from the biostation to the bird box at junction d and back to the biostation. If it is not possible for Ollie to complete the trip, output `impossible`.

Sample Input 1	Sample Output 1
4 3 90 90 3 2 3 45 3 2 0 4 2 315 2 1 3 135 3 2 270 3 1 2 180 2 2 90 4 2 225 2 1 2 135 3 2 270	7

Sample Input 2	Sample Output 2
2 2 90 90 1 2 10 0 1 1 15 180	impossible

Problem C
Cheese, If You Please

The "We Cut The Cheese" specialty food store sells specialized blendings of cheeses. For example, their Italian Blend is made up of 50% provolone, 30% mozzarella and 20% parmesan, while their African Safari is made up of 74% domiati, 25% areesh with just a hint (1%) of bokmakiri. You started working at the store as a cheese blend taster, but two years and 45 pounds later you have worked your way up to head of the accounting office. Every week the store gets various shipments of cheeses, depending on the time of year, market price and other factors. Given these amounts and the percentages required for each cheese blend, you are asked every week to determine the optimal use of these cheeses to maximize profit. When the store just made a few different cheese blends this could be done by hand, but with business expanding faster than a cheese souffle, the number of blends has also grown to the point where a program is now needed to determine the optimal use of the cheese shipments. So the question is: Are you gouda-nough to write such a program?

Input

Input starts with a line containing two positive integers n m ($1 \leq n, m \leq 50$), where n is the number of types of cheese used to make the cheese blends and m is the number of different cheese blends offered by the store. The next line contains n integers w_1 w_2 ... w_n ($0 \leq w_i \leq 500$), where w_i is the number of pounds of cheese type i that the store has on-hand. Following this are m lines of the form p_1 p_2 p_3 ... p_n t ($0.0 \leq p_i \leq 100.0$, $0.0 \leq t \leq 10.0$), where p_i indicates the percentage of cheese i found in the blend, and t is the profit per pound for the blend.

Output

Output the maximum profit that can be obtained for the given pounds of cheese, blending percentages and profits, assuming all of the blended cheeses get sold. Round your answer to the nearest penny.

Sample Input 1	Sample Output 1
3 2 100 150 100 50.0 50.0 0.0 3.20 0.0 50.0 50.0 2.80	920.00

Sample Input 2	Sample Output 2
3 2 100 150 100 50.0 50.0 0.0 3.20 0.0 40.0 60.0 2.80	1000.00

This page is intentionally left blank.

Problem D
Follow the Bouncing Ball

There are several games you can download on your phone that are variations of the following idea. The screen is filled with various convex geometric objects each with a number inside. At the bottom of the screen is a gun which fires n balls in a rapid-fire manner when its trigger is pulled, each ball leaving the gun 1 second after the previous one. The gun swivels to allow you to set the direction for the balls. Once they leave the gun the balls will bounce off the geometric objects and the walls until they return to the bottom of the screen at which time they disappear. The balls do not interact with each other—if two balls intersect they simply pass through each other unimpeded.

Each time a ball hits one of the geometric objects, the number inside the object decrements by 1. The moment an object's count reaches zero it disappears and the ball (or balls) that just hit it continues traveling without bouncing off the object. Consider the example in Figure D.1: if the gun (indicated by the letter G) is facing to the left at a 45 degree angle then all the balls will follow the zigzag path starting at G. Note that the rectangular object can be hit twice by any one ball – once from the left and once from the right. Once this object is hit 10 times by any combination of left and right hits it will disappear and the paths of all balls on the screen will change accordingly.

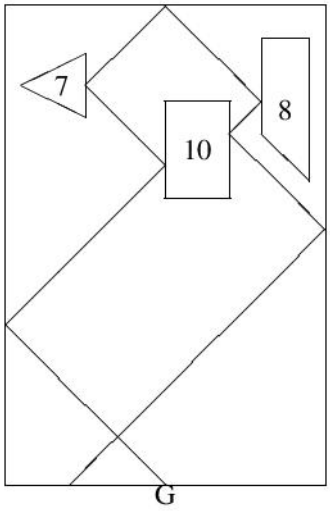

Figure D.1: Sample Input 1.

In the actual games the goal is to get rid of all the objects as quickly as possible. Here we're just interested in one sequence of balls fired from a fixed gun angle. More specifically, given the locations of a set of objects, the numbers inside of them, the number of balls shot and the orientation of the gun, determine what the final values will be in each of the objects.

Input

Input begins with seven values w h n m l r s (all integers except for l). The first four denote the width and height of the screen ($1 \leq w, h \leq 1\ 000$), the number of balls shot by the gun ($1 \leq n \leq 500$), and the number of objects on the screen ($1 \leq m \leq 20$). The last three describe the location of the gun along the bottom edge of the screen ($0 \leq l \leq w$) and its orientation, as shown in Figure D.2 ($-10 \leq r \leq 10$, $1 \leq s \leq 10$).

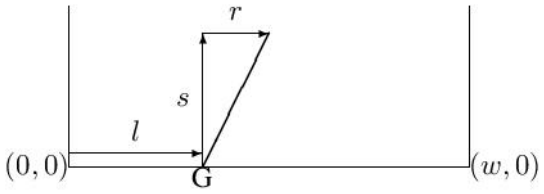

Figure D.2: Gun location and orientation.

Following this are m lines describing the m convex objects. Each of these lines starts with an integer p indicating the number of edges of the object ($3 \leq p \leq 10$), following by p x-y coordinates of the edge endpoints, given in clockwise order. Following this is an integer q indicating the value inside the object ($1 \leq q \leq 1\ 000$). All lengths are in centimeters and the balls leave the gun traveling at 1 centimeter per second. No two objects (including the boundary of the screen) intersect each other, not even at a point. Objects are removed at the instant their values hit 0 or less. Any ball that hits an object within 10^{-7} second before its removal time will pass through the object as if it were removed. Finally, no ball will strike an edge within 10^{-7} of one of the edge's endpoints and all balls will eventually leave the game screen.

Output

Output m values indicating the final values in each of the m objects, in the order that they appear in the input. If any of these values are negative just output 0 instead.

Sample Input 1

```
20 30 5 3 10 -1 1
4 10 18 10 24 14 24 14 18 10
3 5 23 1 25 5 27 7
4 16 22 16 28 19 28 19 19 8
```

Sample Output 1

```
0 2 3
```

Sample Input 2

```
20 30 10 3 10.1 -1 1
4 10 18 10 24 14 24 14 18 10
3 5 23 1 25 5 27 7
4 16 22 16 28 19 28 19 19 8
```

Sample Output 2

```
0 0 1
```

Problem E
Just Passing Through

Justin and Fred are taking a road trip, traveling from West to East across their state. They have a few Road Trip Rules:

1. They must have an awesome time!
2. The trip must begin somewhere along the Western edge of the state and finish somewhere along the Eastern edge of the state.
3. Each step in their trip must move them either directly East, diagonally Northeast, or diagonally Southeast.
4. They want to travel through exactly n "passes" (defined below).
5. Due to Fred's sensitivity to higher elevations, they wish to minimize the cumulative sum of elevations during the trip.
6. They must have an awesome time!

Because Justin and Fred are traveling Eastward, a "pass" is any location with strictly lower elevations to its East and West and strictly higher elevations to its North and South. Consider the elevation map shown in Figure E.1. Undrivable locations (either due to water or stubborn insistence on staying in-state) are shown in black, while passes are shown as gray. Locations adjacent to the border or to undrivable locations are not eligible for considerations as passes.

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

Figure E.1: A sample elevation map

Input

The input begins with three integers $r\ c\ n$ representing the number of rows and columns in the representation of the state's topography ($3 \leq r, c \leq 500$) and the exact number of passes to be crossed ($0 \leq n \leq 10$). The next r lines each contain c elevation entries. Undrivable locations are represented by -1, and all other elevations are between 0 and $1\,000$. There is guaranteed to be at least one drivable location on both the Eastern and Western borders.

Output

Output the sum of the elevations along the optimal path which satisfies the Road Trip Rules. If no such path exists, output `impossible`.

Sample Input 1

```
5 7 2
-1 -1 2 5 4 3 1
3 4 1 4 1 2 1
3 4 5 5 3 4 5
2 3 2 1 2 3 2
-1 5 4 1 4 4 2
```

Sample Output 1

```
14
```

Sample Input 2

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

Sample Output 2

```
impossible
```

Problem F
Musical Chairs

Professor O'Dagio of the music department at Faber College has come up with a rather interesting way of selecting its department chair. All n members of the music faculty line up, then the first one in line calls out an integer k corresponding to the opus number of his or her favorite musical composition by Faber College's most illustrious alumnus, composer I. M. Tondeff. The department members then "count off," starting with the first in line and cycling back to the beginning of the line if necessary. When the count reaches k, that person steps out of the line and is relieved (in more than one sense!) of chairmanship duty for that year.

The next person in line then calls out his or her favorite opus number (this becomes the new value of k) and the count restarts at "1," and continues until the next person is eliminated, and so on. When only one faculty member is left standing, this is the new department chair. To prevent cheating, everyone's favorite number is announced in advance and no one is allowed to choose Tondeff's Opus 1 (the famous drinking song *Rhapsody in Brew*).

For instance, suppose the professors are numbered 1 through 4 and lined up in that order; suppose their favorite opus numbers are, respectively, opus 8 (*The Four Sneezings*), opus 2 (*Concerto for Kazoo and Cigar Box Banjo*), opus 4 (*The Taekwondo Rondo*), and opus 2 (again). Figure F.1 shows the process by which the new chair is selected.

(1)	(2)	(3)	(4)	
8	2	4	2	Professor (1) calls out "8" and begins counting
(1)	(2)	(3)		
8	2	4		Professor (4) is eliminated. Professor (1) (the next in line) calls out "8" and begins counting
(1)		(3)		
8		4		Professor (2) is eliminated. Professor (3) (the next in line) calls out "4" and begins counting
		(3)		
		4		Professor (1) is eliminated. Professor (3) is the new department chair

Figure F.1: Example of Selection Process

Input

The first line of input contains an integer n ($2 \leq n \leq 10^4$), the number of faculty members. The next line contains n integers $k_1 \ldots k_n$ ($2 \leq k_i \leq 10^6$ for each i), where k_i is the favorite opus number of professor i.

Output

Output the number of the new department chair.

Sample Input 1	Sample Output 1
4 8 2 4 2	3

Problem G
Out of Sorts

Ann Logan is fascinated with finite sequences of integers. She is particularly interested in sequences of the form $x_1, x_2, \ldots, x_n$ where:

- $x_i = (ax_{i-1} + c) \mod m$,
- n, m, a, and c are positive integer constants,
- x_0 is a non-negative integer constant, and
- all n values are unique.

For example, if $n = 5$, $m = 8$, $a = 1$, $c = 3$, and $x_0 = 3$, the sequence is 6, 1, 4, 7, 2 ($x_1 = (1 \cdot 3 + 3) \mod 8 = 6$, $x_2 = (1 \cdot 6 + 3) \mod 8 = 1$, and so on). Note that she does not consider the initial value x_0 to be part of the sequence.

Ann wants to be able to quickly determine, for any integer value, whether or not it appears within a finite sequence of this form. Given values of n, m, a, c, and x_0, she plans to follow this list of steps:

1. Generate the sequence $x_1, \cdots, x_n$ and store it in an array.
2. Sort the array.
3. Perform a binary search of the array for each integer of interest to her.

Ann's search algorithm, while not the most efficient possible, is pretty straightforward and understandable to anyone familiar with binary search: after calculating the midpoint `mid` at each step of the calculation (using `mid = (low+high)/2`), she first checks whether or not the value at location `mid` is equal to the search value `x`. If not, she then narrows the search according to whether `x` is strictly less than or strictly greater than the value at location `mid`.

Unfortunately, Ann is absent-minded and she lost her list of steps. She managed to remember the first and last step, but . . . she forgot to sort the array before performing her binary search! Needless to say, this means that many values that are in the (unsorted) array will not be found by a binary search, although surprisingly some can. In the example above, both 4 and 7 can be found with Ann's binary search. How many values can be found for various sequences? Don't botch it up!

Input

Input consists of a line containing five integers n, m, a, c, and x_0 ($1 \leq n \leq 10^6$, $1 \leq m, a, c \leq 2^{31} - 1$, $0 \leq x_0 \leq 2^{31} - 1$), where n is the length of the sequence $x_1, \ldots, x_n$ to be generated and m, a, c, and x_0 are the constants used for generating the sequence. All values in the generated sequence are guaranteed to be unique.

Output

Output the number of sequence values that could be found using Ann's version of binary search, assuming she forgot to sort the sequence.

Sample Input 1	Sample Output 1
5 8 1 3 3	2

Sample Input 2	Sample Output 2
6 10 1234567891 1 1234567890	6

Problem H
Remainder Reminder

"Hey Bill, you were in charge of shipping those novelty book remainders to Remainderville, OH, right?" asked Fred. "Those were cute little books weren't they, each shaped like a cube one inch on each side. I can't understand why they didn't sell better. Can you remind me how many we shipped?"

"Well," started Bill, "I can't remember the number exactly. I do remember that we tried three different size boxes. When we used the largest box size we had 407 books remaining after filling all of the rest completely. When we tried the second largest box size we had 409 books left over, and when we tried the smallest box size we only had 17 books left over."

"Okay," a slightly puzzled Fred continued, "then just let me know the box sizes and the number of boxes you used."

"Funny thing" replied Bill, "I can't remember that either. I do remember that the boxes were made from a 16 by 21 inch sheet of cardboard, with squares cut out at each of the four corners. We then folded the sides up to make an open top box, filled each box with books, then used packaging tape to attach a lid. The dimensions of each box were all integer multiples of inches."

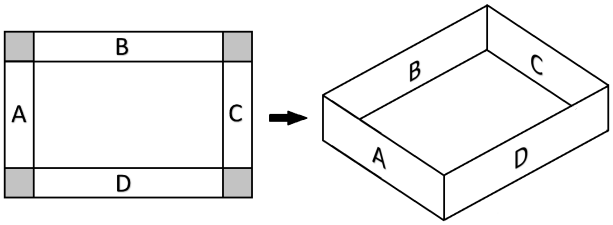

Figure H.1: Example of folding a box.

"You seem to have a very selective memory," a now aggravated Fred complained.

"Well, if it helps, I know that the three different box sizes were the three largest possible to make from those sheets of cardboard. I also know that we had between $20,000$ and $30,000$ books to ship. I think that should give you enough to determine the total number of books."

"So let me get this straight. You're giving me the dimensions of the sheets of cardboard, the fact that the box sizes are the three largest that can be made from them, the number of books left over after using each of the box sizes, and a range in which the number of books lies, correct?" said Fred, rubbing his increasingly aching head.

"That's it," confirmed Bill.

"Sounds like some grossly convoluted computer programming problem!"

"Yeah. Go figure?"

Input

Input consists of seven positive integers a b c d e f g, where a and b ($a \leq b$, $7 \leq a, b \leq 100$) are the dimensions of the sheets of cardboard (in inches), c, d and e ($1 \leq c, d, e \leq 10^9$) are the number of books left over for the three largest size boxes possible for the given size cardboard sheets (in order from largest box size to third largest box size) and f and g specify an inclusive range for the number of books ($1 \leq f < g \leq 10^9$).

Output

Output the number of books which satisfy all the conditions of the problem. Each problem is guaranteed to have a unique answer.

Sample Input 1	Sample Output 1
16 21 407 409 17 20000 30000	22457

Problem I
Square Rooms

Bob Roberts is an archaeologist who specializes in the buildings of an ancient race known as the Erauqs. This ancient race was known not only for the fabulous treasures they amassed over the years, but also for the peculiar room layouts of their buildings. Their buildings were always rectangular and the rooms inside the buildings were always square shaped. (To preserve this property, some areas of the building were filled with solid rock.)

When Bob comes across evidence of a buried Erauqi building, he first uses an ultrasound device to find out as much about the layout of the building as possible before he starts digging. The ultrasound can tell him the location of any treasure and the rock locations, but it is not accurate enough to determine the locations of the walls. However, Bob knows that wall locations can be determined from the fact that the Erauqs always put exactly one treasure in each square room. For example, if the ultrasound were to show the locations of treasures and rock as shown in the left of Figure I.1, then the only possible layout of the rooms would be as shown on the right.

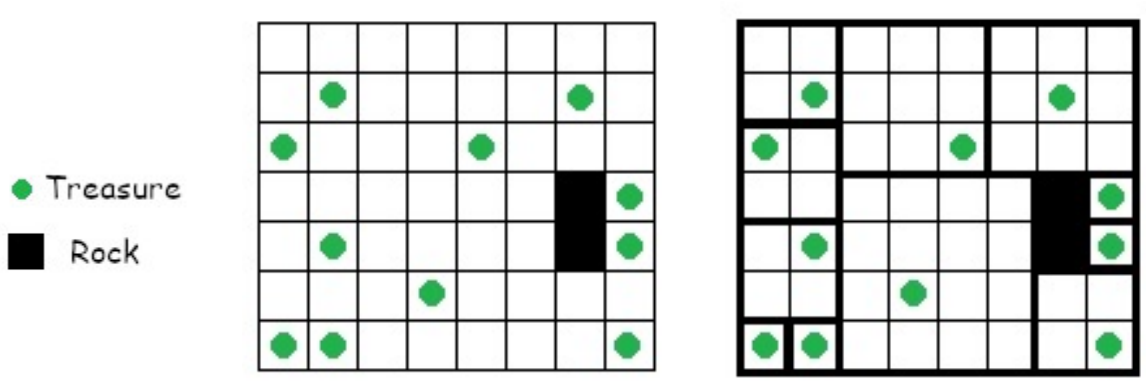

Figure I.1: Ultrasound results and wall locations

Occasionally, Bob comes across a building by a rival tribe, the Elgnatcer. In these cases, it is not possible to find room layouts that are compatible with Erauqi architectural principles.

Bob's come to you to write a program to locate the walls of the square Erauqi treasure rooms and to identify the non-Erauqi buildings. Frankly, the problem has been driving him up a wall.

Input

Input starts with a line containing two integers n m ($1 \leq n, m \leq 100$), the number of rows and columns in the building grid. Following this are n lines each containing m characters. These characters are either '`$`', '`#`' or '`.`' for treasure, rock or empty space. There are at least 1 and no more than 52 treasures.

Output

For each test case, if there is no way to assign square rooms to all of the empty spaces in such a way that each room contains one treasure, output `elgnatcer`. Otherwise, output an n by m grid indicating the layout of the Erauqi building. Assign to each square room a letter, starting with '`A`', then '`B`', '`C`', ..., '`Z`', '`a`', ..., '`z`'. Room labels are assigned by moving left to right across each row, starting at the topmost row (the first input row for the test case), and assigning the next room label to each unlabeled room as it is encountered. Each grid square in the output should contain either the corresponding room label or the character '`#`' indicating rock.

Sample Input 1

```
7 8
........
.$....$.
$...$...
......#$
.$....#$
...$....
$$.....$
```

Sample Output 1

```
AABBBCCC
AABBBCCC
DDBBBCCC
DDEEEE#F
GGEEEE#H
GGEEEEII
JKEEEEII
```

Sample Input 2

```
1 5
#$#$#
```

Sample Output 2

```
#A#B#
```

Sample Input 3

```
1 5
#$.$.
```

Sample Output 3

```
elgnatcer
```

Problem J
Taxed Editor

Reed Dacht is a freelance book editor who works out of his home. Various publishers send him texts to read along with a deadline date for when they would like his editorial comments to be sent back to them. Reed is very precise with his reading and will never read more than one book at a time, so he never starts a new book until after he is finished reading the previous one. He is willing to change the rate (number of pages per day) at which he reads books, but to be fair to each of his clients he insists on using the same speed for all books.

The other day Reed got a large set of editing requests and has a bit of a problem. He doesn't think that there's any way he can read all of the books by the specified deadlines. He could set his reading speed high enough to get them all done in time, but that might compromise the accuracy of his editing (as well as severely tax his eyesight). To avoid that he could select a slower reading speed, but then too many of the editing jobs would go past their deadlines. He has decided on a compromise: he is willing to let a small number of books go past deadline (so as to annoy only a small number of clients) and will set his reading speed to the smallest possible number of pages per day so that no more than that number of books will be late. But what should that reading speed be? That's where you come in.

Input

Input starts with two integers n m where n ($1 \leq n \leq 10^5$) is the number of books to edit and m ($0 \leq m < n$) is the number of editing jobs that Reed will allow to be late. Following this are n lines each describing one book. Each line contains two integers l d where l ($1 \leq l \leq 10^9$) is the length (in pages) of the book and d ($1 \leq d \leq 10^4$) is the deadline for that book (number of days until is it due).

Output

Output the minimum integer speed (pages per day) which allows no more than m late editing jobs.

Sample Input 1	Sample Output 1
3 1 450 9 500 6 300 4	84

This page is intentionally left blank.

Problem K
Where Have You Bin?

Ben Bean owns Ben Bean's Bin Bonanza which offers storage bins to the five big companies in town: a rubber band company, a Mercedes Benz dealership, a bing cherry farm, a bonbon candy store and a bun bakery. Together they store their bands, Benzes, bings, bonbons and buns in Ben's bins.

Ben has n bins altogether arranged in a single row, numbered 1 to n. While not all of the bins may be in use at any given time, Ben finds it useful to keep all the bins for one company contiguous. Thus, when a company needs a new bin or relinquishes one it no longer needs, Ben may need to move items from one bin to another to make sure all of that company's bins stay next to each other. Sometimes Ben has a choice of which bin to move, so he has assigned a cost to each bin equal to the number of items stored in the bin (removing items from a relinquished bin and/or moving items into a new bin are the company's responsibility and do not add to Ben's costs). Obviously when moving bins Ben wants to keep the cost as low as possible.

If a single company has to make changes Ben can usually figure out the cheapest way to move items around, but typically at the end of each business quarter all of the five companies will make additions and deletions of bins as they reassess their product lines. In cases like this, deciding the lowest-cost set of moves is more difficult. Consider the situation in Figure K.1a which shows 6 bins storing items from the five companies labeled A, E, I, O and U. The numbers in parentheses indicate the number of items in that bin (and thus the cost of moving the items from that bin to another). Suppose that at the end of the quarter company U decides it no longer needs bin 6 and company A requests a second bin. One possibility is to move E's items from bin 2 to the empty bin 6 which frees up bin 2 for company A (Figure K.1b)—the cost of this rearrangement to Ben is 4. The optimal move though is to move U's items from bin 5 to bin 1 and move A's items from bin 1 to bin 5 and giving company A bin 6 (Figure K.1c)—the cost of this move is 3. Ben could also have moved A's items from bin 1 to bin 6 to achieve the same optimal cost. Notice that in all cases removing the three items from U's bin 6 and adding the seven items to A's second bin does not cost anything to Ben.

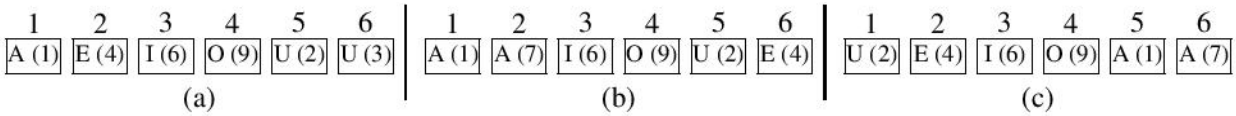

Figure K.1: Sample Input 1.

Input

Input starts with a string of characters of length n ($1 \leq n \leq 150$) indicating the initial usage of the bins. The characters will all be from the set {A, E, I, O, U, X} indicating either the company using the bin or an empty bin (X). Following this is a row of n integers indicating the number of items in each bin; values at locations corresponding to an empty bin will always be 0 and values at locations corresponding to a company's bin will be positive and ≤ 100. Bins for any one company will always be contiguous.

The next line starts with an integer d ($0 \leq d \leq n$) indicating the number of deletions for this quarter. Following this are d integers. Each integer specifies a bin which is no longer needed by a company. None will ever correspond to an already empty bin. On the final line is a positive-length string of characters indicating

the new bins requested. If this string is a single X then no new bins are being requested. Otherwise the characters will all be in the set {A, E, I, O, U}, in no particular order, each character representing a request for a new bin by the corresponding company. There will always be enough bins to handle any set of changes.

Output

Output the minimal cost required to satisfy all the bin changes while keeping each company's bins contiguous.

Sample Input 1	Sample Output 1
AEIOUU 1 4 6 9 2 3 1 6 A	3

Sample Input 2	Sample Output 2
AEIOUU 10 4 6 9 2 3 1 6 A	4

Sample Input 3	Sample Output 3
AEIOUU 1 4 6 9 2 3 4 5 1 4 3 EUE	0