German Collegiate Programming Contest 2025

July 5th

Problems

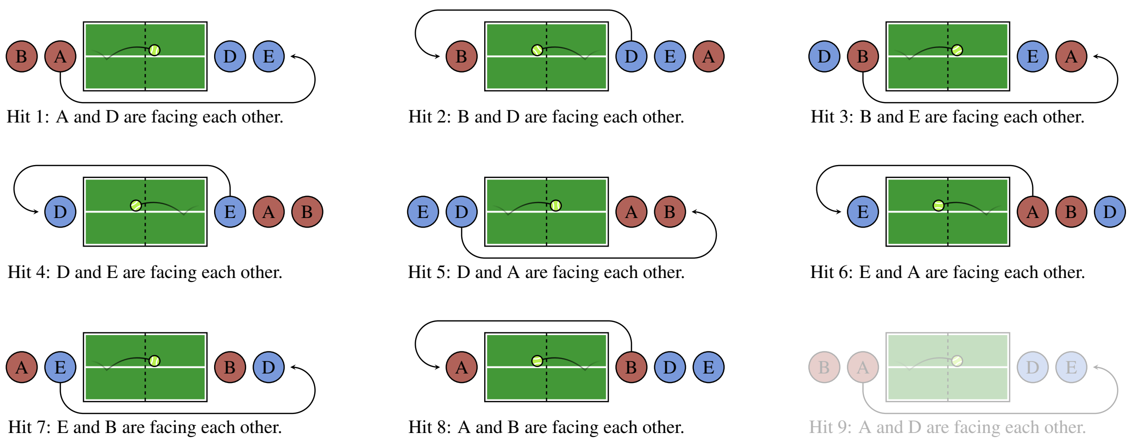
This page is intentionally left (almost) blank.

Problem A: Around the Table

Time limit: 1 second

Emilia and her brother Alex meet up with a lot of friends in the park to play table tennis together. Since there is only one table, they play round-the-table which is everybody's favourite.

The rules are simple: There are $\ell \geq 2$ kids queueing up on the left side and $r \geq 1$ kids queueing up on the right side of the table. The first kid on the left side starts with the ball. Whoever has the ball hits it over the net onto the opposite side and runs around the table to join the queue at the opposite side. This is repeated until the first mistake.



B
A
D
E
Hit 1: A and D are facing each other.
B
D
E
A
Hit 2: B and D are facing each other.
D
B
E
A
Hit 3: B and E are facing each other.
D
E
A
B
Hit 4: D and E are facing each other.
E
D
A
B
Hit 5: D and A are facing each other.
E
A
B
D
Hit 6: E and A are facing each other.
A
E
B
D
Hit 7: E and B are facing each other.
A
B
D
E
Hit 8: A and B are facing each other.
B
A
D
E
Hit 9: A and D are facing each other.

Figure A.1: Visualization of the first sample. During Hit 1 and 5, and during Hit 3 and 7, we observe the same pairing. After 8 hits we end up in the initial state again and cannot observe any new pairing afterwards. Therefore, there are 6 different pairings we will see in total.

Emilia, Alex, and their friends play this game all the time and are so good that they can go on almost forever without making any mistake. After some time, Emilia notices that she has never faced Alex, meaning that they have never been at the front of opposite queues at the same time. She wonders if this will ever happen in this round. Curious about this, she starts keeping track which pairings have already faced each other.

How many different pairings has she counted after the ball went over the net $10^{10^{100}}$ times?

Input

The input consist of:

- One line with two integers ℓ and r ($2 \leq \ell \leq 10^9$ and $1 \leq r \leq 10^9$), the number of players on the left and right side of the table, respectively.

Output

Output the number of different pairings Emilia counts.

Sample Input 1	Sample Output 1
2 2	6

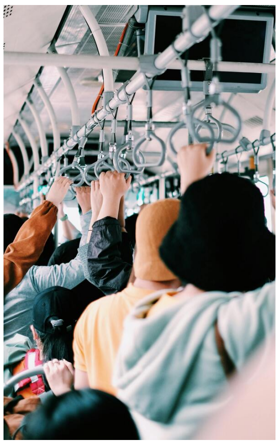
Sample Input 2	Sample Output 2
`2 3`	`10`

Sample Input 3	Sample Output 3
`3 2`	`5`

Sample Input 4	Sample Output 4
`5 2`	`14`

Problem B: Bustling Busride

Time limit: 5 seconds


A crowded bus. By: Pexels/Pew Nguyen

The university of Bithampton is served by exactly one bus line. On its way to the city centre, it serves several stops at which students may exit. Every student has a fixed bus stop where they want to exit.

It is Friday afternoon, 4 PM, and as always, all the students want to go home, leading to quite a long queue at the bus stop. Fortunately, the bus line is served in regular intervals, with the first bus arriving at 4 PM.

Whenever a bus arrives at the university, everyone in the queue tries to enter the bus, which makes the buses very crowded. This has led to numerous complaints where people tried to exit buses but were unable to because of the sheer amount of people. As a consequence, the bus company decided that at every stop which someone in the bus has as destination, everyone in the bus must exit it. Those who want to travel further enter again. For every time a passenger enters or exits the bus, the bus needs to wait w seconds.

To offer the best service, the bus company wants to minimize the maximum time it takes anybody from 4 PM until they reach their destination. For each bus, the bus driver can decide how many people from the front of the queue enter the bus. The number of people that can enter a bus is unlimited. Help the bus drivers make the optimal decisions to achieve the company's goal.

Input

The input consists of:

- One line with four integers n, b, r, and w ($1 \leq n, b \leq 10^5$, $1 \leq r, w \leq 10^6$), the number of passengers, the number of bus stops, the time between the arrival of two buses at the university, and the delay for exiting and entering.
- One line with b integers d_i ($1 \leq d_i, \sum d_i \leq 10^6$), the travel time from the $(i-1)$th bus stop to the ith bus stop (the 0th bus stop is the university).
- One line with n integers t_i ($1 \leq t_i \leq b$), indicating that the destination of the ith person in line is the t_ith bus stop.

All times are given in seconds.

Output

Output the minimum number of seconds until every person in line has exited at their destination.

(Samples start on the next page)

Sample Input 1

```
3 3 20 1
2 2 2
2 3 1
```

Sample Output 1

```
18
```

In this example, it is optimal to let everyone board the first bus. The first boarding takes 3 seconds. Then, the bus drives for 2 seconds to get to the bus stop 1. After 3 seconds, everybody exited and after another 2 seconds, the people continuing their journey are back on the bus. After 2 more seconds, they arrive at the next stop where it takes 2 seconds for everybody to exit and one second for the remaining person to get back on the bus. After 2 more seconds, the bus arrives at the final destination where it takes one second for the last person to exit.

Sample Input 2

```
4 2 1 10
3 2
1 2 2 1
```

Sample Output 2

```
27
```

It is optimal to use one bus per person.

Sample Input 3

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

Sample Output 3

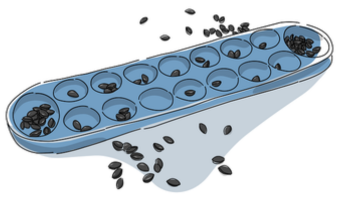
```
17
```

It is optimal to assign the first two passengers to one bus. Everybody else gets their own bus.

Problem C: Congklak

Time limit: 2 seconds

Alice and Bill really enjoy playing board games, and they are always looking for new challenges. Recently, they discovered the Indonesian game *Congklak* which is played with a game board made up of several holes containing some number of stones. After playing some games, Alice quickly got the hang of it and won every game, so Bill did not want to play any more. Instead, inspired by the rules of the game, he came up with the following challenge for Alice:


A typical Congklak game board.
CC BY-SA 4.0 by rikadiani17 on Pixabay

There is a game board with n holes arranged in a long row. These holes are numbered from 1 to n from left to right. Initially the ith hole contains a_i stones. Note that this setup differs from the usual Congklak game, where the game board consists of two rows and one large hole at each end.

Now Bill will play t games where each game goes as follows:

Bill starts the game at the first hole, holding one new stone in his hand. He then moves along the game board from hole 1 to hole n. At each hole i, he first checks how many stones are currently in the hole, and depending on the result he performs exactly one of the following two actions:

i) If the hole is empty, he drops one stone into it. Next, he checks how many stones are still in his hand. If his hand is empty, the game stops. Otherwise, he moves his hand to hole $i + 1$ next and repeats the steps.

ii) If there is at least one stone in the hole, he also drops one stone into it. Next, he checks how many stones are still in his hand. If his hand is empty, he takes out all the stones from hole i into his hand. Regardless of whether or not his hand was empty, he moves his hand to hole $i + 1$ next and repeats the steps.

When Bill moves his hand past hole n, the game stops and Bill discards any stones that he still holds in his hand.

Bill challenges Alice to predict in advance the number of stones in every hole after playing exactly t games. Note that the game board is **not** reset after playing a game, i.e. the initial configuration of the second game is the same as the configuration when the first game ends.

Input

The input consists of:

- One line with two integers n and t ($1 \leq n \leq 10^5$, $1 \leq t \leq 10^{12}$), the number of holes and the number of games.
- One line with n integers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^{12}$), where a_i describes the initial number of stones in the ith hole.

Output

Output n integers, the ith of which is the number of stones in hole i after playing t games.

Sample Input 1

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

Sample Output 1

```
0 4 0 1 2 1 5
```

In the first sample, Bill plays exactly one game. The figure below visualizes the steps performed during this game.

Sample Input 2

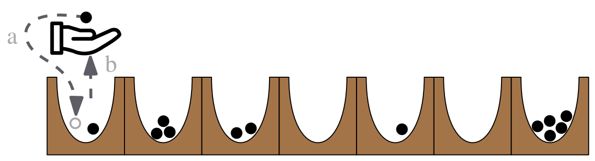
```
4 4
1000000000000 1 2 3
```

Sample Output 2

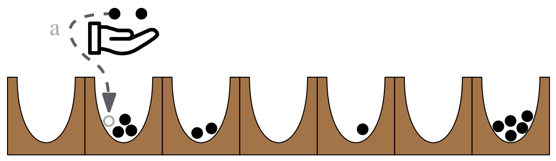
```
1 3 0 5
```

In the second sample, the initial number of stones per hole is $[1000000000000, 1, 2, 3]$. The number of stones per hole after each game is:

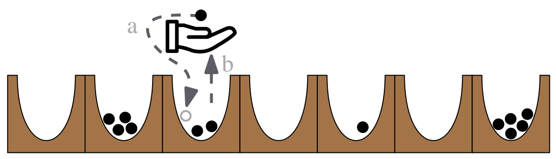
1. $[0, 2, 3, 4]$
2. $[1, 2, 3, 4]$
3. $[0, 3, 0, 5]$
4. $[1, 3, 0, 5]$



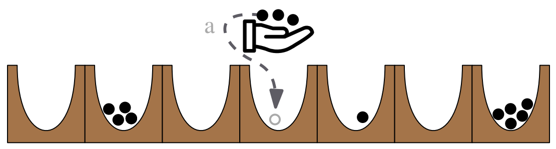
1. Hole 1 is not empty (case ii).
   (a) Bill drops one stone into hole 1.
   (b) Bill's hand is empty now, so he takes all two stones out of hole 1 and continues to the next hole.



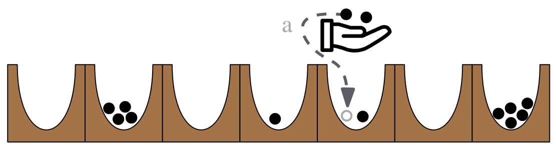
2. Hole 2 is not empty (case ii).
   (a) Bill drops one stone into hole 2.
   (b) Bill still holds one stone, so he just continues to the next hole.



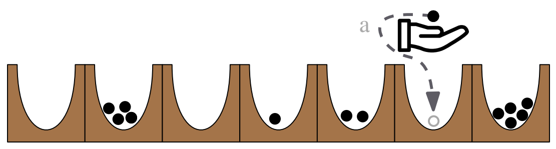
3. Hole 3 is not empty (case ii).
   (a) Bill drops one stone into hole 3.
   (b) Bill's hand is empty now, so he takes all three stones out of hole 3 and continues to the next hole.



4. Hole 4 is empty (case i).
   (a) Bill drops one stone into hole 4.
   (b) Bill still holds two stones, so he just continues to the next hole.



5. Hole 5 is not empty (case ii).
   (a) Bill drops one stone into hole 5.
   (b) Bill still holds one stone, so he just continues to the next hole.



6. Hole 6 is empty (case i).
   (a) Bill drops one stone into hole 6.
   (b) Bill's hand is empty now, so the game stops.

Figure C.1: Visualization of the only game which Bill plays in the first sample.

Problem D: Demand for Cycling

Time limit: 1 second

In the Grid City of Perpendicular Cycling (GCPC for short), cars have never been popular. Instead, everyone uses various kinds of bicycles to make their way around the city. This is not the only feature of GCPC. Following an ancient city design used in different places since at least 2600 BC, all streets are going in one of the cardinal directions (i.e. either north/south or east/west) and therefore only meet at right angles.

The city council is planning on adding a new bike path surrounding the entire city to accommodate the strong demand for good cycling infrastructure. This new bike path should respect the historic grid system and should therefore only consist of segments parallel to one of the cardinal directions.

In order to minimize the construction cost and also the travel time for cyclists, the council wants this new bike path to be as short as possible. The council already prepared the outline of the city but now needs help in finding such a shortest bike path. For simplicity, you may assume that the bike path has width zero and may have distance zero to any point on the outline of the city.

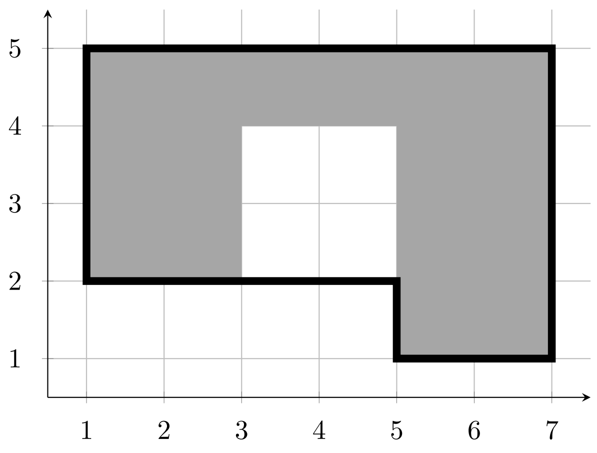


Figure D.1: Illustration of the first sample. The gray area represents the city and the black contour an optimal bike path. The given bike path has length 20. It can be shown that there is no shorter one that satisfies the requirements.

Input

The input consists of:

- One line with an integer n ($4 \leq n \leq 10^5$), the number of points of the outline of the city.
- n lines, each with two integers x and y ($1 \leq x, y \leq 10^9$), the coordinates of the points.

Additionally, the following is guaranteed:

- The points of the city's outline are given in counterclockwise order.
- Two consecutive points have either the same x-coordinate or the same y-coordinate.
- No three consecutive points are collinear.
- The outline of the city does not intersect itself.

Output

Output an integer m ($m \leq n$), the number of points of the bike path, followed by m lines containing the coordinates of the bike path in counterclockwise order. Note that two consecutive points must have either the same x-coordinate or the same y-coordinate, and that no three consecutive points may be collinear.

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

Sample Input 1	Sample Output 1

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

Sample Input 2	Sample Output 2

```
8
1 1
4 1
4 3
3 3
3 4
2 4
2 2
1 2
```
	```
	8
	4 3
	3 3
	3 4
	2 4
	2 2
	1 2
	1 1
	4 1
	```

Sample Input 3	Sample Output 3

```
12
1 3
1 2
2 2
2 1
3 1
3 2
4 2
4 3
3 3
3 4
2 4
2 3
```
	```
	12
	4 3
	3 3
	3 4
	2 4
	2 3
	1 3
	1 2
	2 2
	2 1
	3 1
	3 2
	4 2
	```

Problem E: Engineering Excellence

Time limit: 5 seconds

You are the engineer in charge of designing a wheel for a new space rover. As you do not have enough time to reinvent the wheel, you decide to copy your predecessor's work and make only one small change.

Looking at the plan, you notice that your predecessor's wheel has the shape of a convex polygon for structural reasons. It is well known that wheels with a larger perimeter roll farther per rotation, so surely they must be superior. You attempt to increase the perimeter by as much as possible by moving a single point on the outside of the wheel. While experimenting, you notice that wheels do not seem to work if they are non-convex or if there is an internal angle below 90 degrees.

What is the maximum possible achievable increase in the perimeter of the wheel without violating the above restrictions?

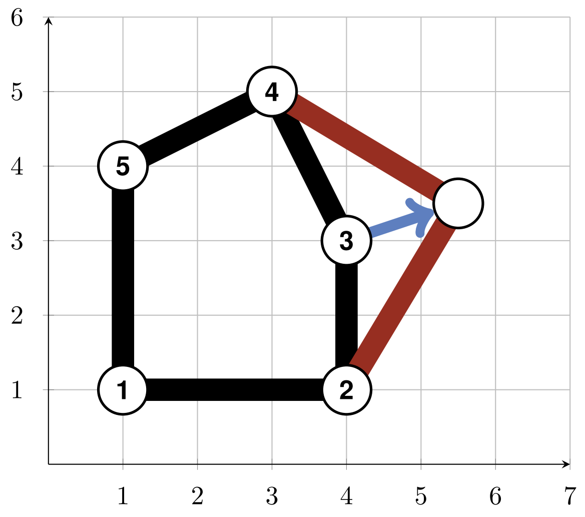


Figure E.1: Visualization of the first sample. Point 3 is moved to $(5.5, 3.5)$, increasing the perimeter by ≈ 1.59488.

Input

The input consists of:

- One line with an integer n ($4 \leq n \leq 10^5$), the number of points of the wheel.
- n lines, each with two integers x and y ($|x|, |y| \leq 10^5$), the coordinates of the points.

The points are given in counterclockwise order and form a convex polygon with no internal angle below 90 degrees. Note that the convex polygon may contain **collinear** points, but no two points are at the same position.

Output

Output the maximum possible absolute increase of the perimeter of the wheel.

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

Sample Input 1

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

Sample Output 1

```
1.594883917346
```

Sample Input 2

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

Sample Output 2

```
3.151142198204
```

Sample Input 3

```
4
0 0
1 0
1 1
0 1
```

Sample Output 3

```
0.000000000000
```

Problem F: Fair and Square

Time limit: 9 seconds

Felix has ordered a large rectangular pizza for his birthday party. He only quickly went to the kitchen to grab some plates and cutlery, and when he came back his guests had already started helping themselves to various parts of the pizza. The pizza, which originally was made up of $h \times w$ square pieces of equal size, is now missing some of these pieces:

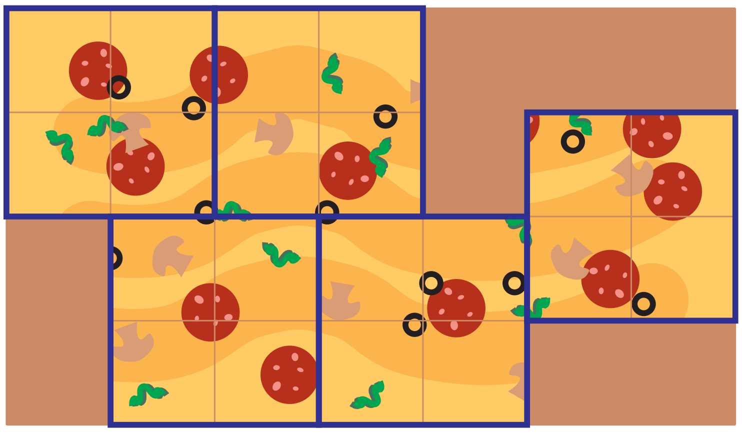


Figure F.1: Illustration of the first sample case, with a division into squares of side length 2.

To ensure that the distribution of the rest of the pizza proceeds in a much more orderly fashion, Felix wants to divide up the remaining pizza into larger square shaped areas. Felix can only cut along the grid lines and cannot rearrange any parts of the pizza. Each square should have the same side length, which should be as large as possible in order to minimize the number of plates needed.

Find this largest possible side length. Note that an answer always exists because the pizza can always be divided into 1×1 squares.

Input

The input consists of:

- One line with two integers h and w ($1 \le h, w \le 2500$), the height and width of the grid.
- h lines, each with a string of length w consisting of '#' (denoting pizza pieces that are still there) and '.' (denoting pizza pieces that have already been taken).

It is guaranteed that the input contains at least one '#'.

Output

Output an integer, the largest possible side length such that the remaining pizza can be divided into squares of that side length.

Sample Input 1	Sample Output 1
4 7 ####... ####.## .###### .####..	2

Sample Input 2

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

Sample Output 2

```
1
```

Sample Input 3

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

Sample Output 3

```
1
```

Sample Input 4

```
5 13
....###......
....###..###.
.#######..###.
.###.....###.
.###.........
```

Sample Output 4

```
3
```

Problem G: Generating Cool Passwords Company

Time limit: 1 second

You have been tasked with creating a list of very secure passwords to be distributed to the users at the *Generating Cool Passwords Company*. Therefore, given an integer n, generate exactly n passwords which each satisfy the following criteria:

- Each password consists of at least 8 and at most 12 printable ASCII characters with code between 33 (`!`) and 126 (`~`) inclusive. See Figure G.1 for an overview of these.
- Each password contains at least one lowercase letter `a-z`, at least one uppercase letter `A-Z`, at least one digit `0-9` and at least one special symbol (any character that is neither a digit nor a lowercase or uppercase letter).

!	33	-	45	9	57	E	69	Q	81	]	93	i	105	u	117
"	34	.	46	:	58	F	70	R	82	^	94	j	106	v	118
#	35	/	47	;	59	G	71	S	83	_	95	k	107	w	119
$	36	0	48	<	60	H	72	T	84	`	96	l	108	x	120
%	37	1	49	=	61	I	73	U	85	a	97	m	109	y	121
&	38	2	50	>	62	J	74	V	86	b	98	n	110	z	122
'	39	3	51	?	63	K	75	W	87	c	99	o	111	{	123
(	40	4	52	@	64	L	76	X	88	d	100	p	112	|	124
)	41	5	53	A	65	M	77	Y	89	e	101	q	113	}	125
*	42	6	54	B	66	N	78	Z	90	f	102	r	114	~	126
+	43	7	55	C	67	O	79	[	91	g	103	s	115		
,	44	8	56	D	68	P	80	\	92	h	104	t	116		

Figure G.1: All the non-whitespace printable ASCII characters. The four relevant character classes are highlighted in different colours.

Of course, the passwords should not be too similar to each other. Specifically, for every pair of passwords from your list it must be true that they are distinct and that moreover it is not possible to get one from the other by inserting, modifying or deleting a single character. Formally, the edit distance of any two passwords must be at least 2.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of passwords to create.

Output

Output n lines, each with one password according to the rules above. The passwords must have pairwise edit distance at least 2. If there is more than one solution, any one of them will be accepted.

Sample Input 1

```
3
```

Sample Output 1

```
haXXor@1337
hunTer2!!!
abcABC123#@$
```

Sample Input 2	**Sample Output 2**

`2`	`3');DRoP`
	`TABLEteams;2`

Problem H: Happy Hookup

Time limit: 2 seconds

It has happened! You found a match! Excitedly, your match Hannah and you have decided to have a morning stroll together. Since the two of you live in different cities, you decide to meetup at a train station.

Getting to your respective cities' train station is no problem. However, the ongoing renovations of the national train network heavily impact the available connections. Some of them are not available at all, and some others are only one-way. This makes it uncertain whether you can meet up at all.

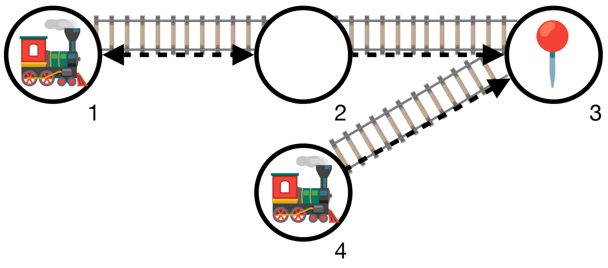


Figure H.1: Visualization of the third sample. Hannah starts from station 4, and you start from station 1. You can both reach station 3 for your meet-up.

You have found a list of the available train connections online. A train station is suitable for your meet-up if both Hannah and you can reach it. Determine whether there is a suitable train station or whether it is necessary to move the morning stroll to another day.

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \leq 10^5$, $0 \leq m \leq 10^5$), the number of train stations and train connections, respectively.
- m lines, each with two integers x and y ($1 \leq x, y \leq n$, $x \neq y$), denoting a direct train connection from train station x to train station y.
- One line with two integers a and b ($1 \leq a, b \leq n$, $a \neq b$), the train stations you and Hannah start from, respectively.

It is guaranteed that no train connection is listed multiple times.

Output

If no suitable train station exists, output "`no`".

Otherwise, output "`yes`", followed by the train station. If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1

3 2
1 3
2 3
1 2
	yes
	3

Sample Input 2

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

Sample Output 2

```
no
```

Sample Input 3

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

Sample Output 3

```
yes
3
```

Problem I: Island Urbanism

Time limit: 7 seconds

On a small volcanic island far away from any major landmass, the *Grand Council for Painless Cycling* (GCPC) is facing regular complaints about the bad quality of the bike path network. Their budget is limited, but they would like to improve the situation for all the cyclists on their island. A survey helped to determine the most common destinations of the cyclists. The GCPC expects that the cyclists are satisfied with the bike path replacements if there is a way to cycle between any two destinations using only newly replaced bike paths. In order not to unduly favour any of the villages, the Council has decided that a maximum of 7 destinations per village should be taken into account.

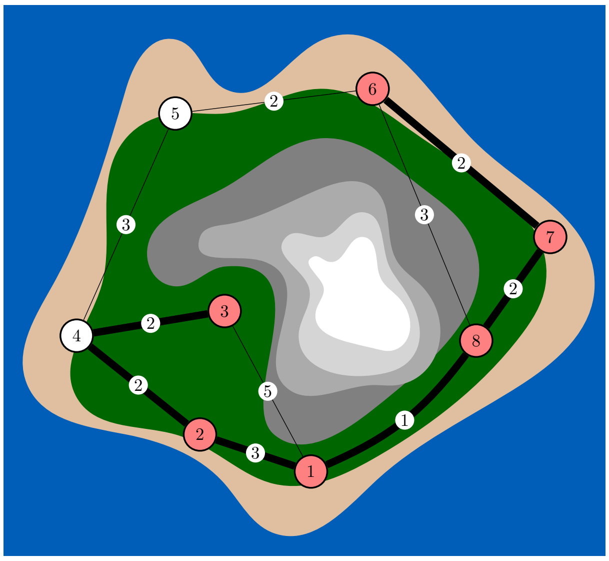


Figure I.1: Illustration of the second sample. Village 1 consists of Junctions 1, 2, 3, and 4, Village 2 consists only of junction 5, and Village 3 consists of the remaining junctions. Junctions that are destinations are coloured red.

There is one main cyclic bike path going around the volcano. On its way, it traverses every village on the island exactly once. Each village may have additional bike paths. How much does the GCPC have to spend at least in order to replace bike paths to connect all destinations via new paths?

Input

The input consists of:

- One line with four integers n, m, v and k ($3 \leq n \leq 5000$, $n \leq m \leq 20\,000$, $3 \leq v \leq n$, $1 \leq k \leq n$), the number of junctions, the number of bike paths, the number of villages, and the number of destinations.
- One line with v integers u_i ($1 \leq u_i$, $\sum_{i=1}^{v} u_i = n$), the ith of which describes the number of junctions in village i.
- m lines each containing three integers a, b and c ($1 \leq a, b \leq n$, $1 \leq c \leq 10^9$), describing a bidirectional bike path between junction a and junction b with a replacement cost of c.
- One line with k distinct integers t ($1 \leq t \leq n$), the junctions that have to be connected by the replaced bike paths.

Additionally, the following is guaranteed:

- Village 1 consists of junctions $1, \dots, u_1$, village 2 consists of junctions $u_1+1, \dots, u_1+u_2$, and so on. The last village consists of junctions $1 + \sum_{i=1}^{v-1} u_i, \dots, n$. Within each village, it is possible to travel between any pair of junctions without leaving the village.
- There is a bike path between junctions $\sum_{i=1}^{j} u_i$ and $1 + \sum_{i=1}^{j} u_i$ for each $1 \le j \le v-1$, as well as a bike path between junctions n and 1. There are no other bike paths between different villages.
- There is at most one bike path between each pair of junctions and no bike path connects a junction to itself.
- In each village there are at most 7 junctions that need to be part of the improved cycling network.

Output

Output the minimum cost which allows replacing bike paths such that every trip between any two destinations can be done using only newly replaced paths.

Sample Input 1

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

Sample Output 1

```
7
```

Sample Input 2

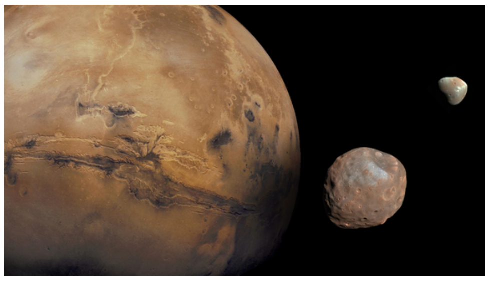
```
8 10 3 6
4 1 3
1 2 3
2 4 2
1 3 5
3 4 2
4 5 3
5 6 2
8 6 3
6 7 2
7 8 2
8 1 1
2 3 1 7 6 8
```

Sample Output 2

```
12
```

Problem J: Jumbled Packets

Time limit per pass: 1 second


Mars and its two moons Phobos and Deimos, Image by NASA's Goddard Space Flight Center / JPL-Caltech / GSFC / Univ. of Arizona on NASA

Due to further and further advancements in technology and space exploration, humanity is sending probes to more planets and moons than ever before. One of these probes is currently residing on Phobos, one of the two Martian moons. This probe transmits binary data back to Earth but it can only do so in short time frames every few hours. Its fast orbit combined with its small size and close proximity to Mars means that Mars blocks radio waves most of the time by physically being in between the probe and Earth. During one time frame of data transmission, one large packet of n binary bits is transmitted, exactly the amount possible within the time frame.

The probe has already been deployed for a few years, so mechanical problems have started to pile up. A recent and very problematic change has been the unreliability of the onboard clock, likely caused by the large and frequent temperature fluctuations ranging from about $-112°$ to $-4°$ Celsius. As a result, the probe has been transmitting data packets at incorrect times, when radio waves cannot reach Earth. To mitigate this, the probe is now sending its data packets multiple times in succession.

Even though this has led to Earth receiving n bits of data, researchers are not able to determine the start and end of the data packet, making the data useless. The received data consists of a possibly empty suffix, followed by a prefix of the sent data, totalling to exactly n bits. For example, if the original packet was "`00101001`", the received data might be "`00100101`" (the colours are only for visual reasons).

You are to write software to encode the data before transmission into a message of the same length, such that you can decode it after receiving the mangled packet. Luckily, the engineering department has managed to improve the design of the receiving antenna, increasing the signal-to-noise ratio so you are now able to use an alphabet of three distinguishable symbols for radio transmission instead of only two.

Your program will be run twice for each test case. In the first pass, your program will be given a binary string to encode. In the second pass, your program will be given your encoded string from the first pass, which may have been modified as explained above. Decode this string to restore the input from the first pass.

A testing tool is provided to help develop your solution.

Input

The input consists of:

- One line with either "`Encode`" or "`Decode`", depending on the action you are to perform.
- One line with an integer n ($1 \leq n \leq 10^5$), the size of the data packet.
- One line with a string s of n characters, the data packet.

In the encoding pass, this string consists of the symbols '`0`' and '`1`'.

In the decoding pass, this string will be a cyclic rotation of your ternary string. In other words, this string is the string you printed in the encoding pass, potentially modified to reflect how this data packet might have been received on Earth, as described in the statement.

Output

In the encoding pass, output the encoded string of length n using the symbols ‘`0`’, ‘`1`’, and ‘`2`’.

In the decoding pass, output the original binary string, the data packet from the encoding pass.

Sample Input 1 (Pass 1)

```
Encode
3
001
```

Sample Output 1 (Pass 1)

```
002
```

Sample Input 1 (Pass 2)

```
Decode
3
200
```

Sample Output 1 (Pass 2)

```
001
```

In the first sample, the encoded ternary string “`002`” was received as “`200`”.

Sample Input 2 (Pass 1)

```
Encode
4
0100
```

Sample Output 2 (Pass 1)

```
2100
```

Sample Input 2 (Pass 2)

```
Decode
4
2100
```

Sample Output 2 (Pass 2)

```
0100
```

In the second sample, the encoded ternary string was received exactly the way it was encoded.

Problem K: Karlsruhe Skyline

Time limit: 1 second

Skyscrapers is a grid logic puzzle in which numbers from 1 to n have to be placed into an $n \times n$ grid. Each number needs to appear exactly once in each row and column. These numbers are to be thought of as skyscrapers which are the respective number of units tall. The rows and columns may have clue numbers on either end which describe the number of visible skyscrapers when looking down that row or column from that position, where taller buildings block the view of any shorter buildings behind them.

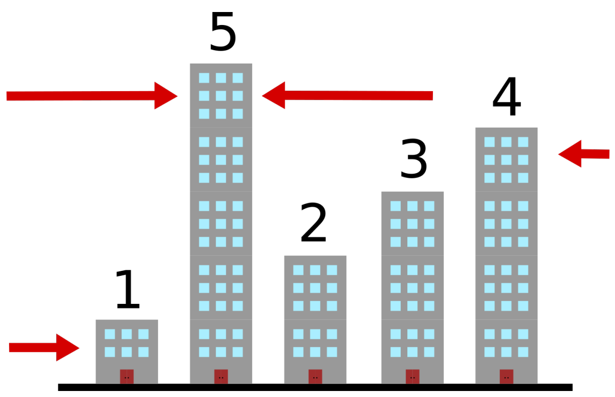


Figure K.1: Illustration of Sample Output 1. Two buildings (1 and 5) are visible from the left and two buildings (4 and 5) are visible from the right.

In this problem we consider only a single row of a Skyscrapers grid which has clue numbers on both ends. Find out whether it is possible to place the skyscrapers from 1 to n in this row to satisfy both clues, and if so, find a valid placement.

Input

The input consists of:

- One line with three integers n, a and b ($2 \leq n \leq 1000$, $1 \leq a, b \leq n$), the length of the row and the clues on the left and right.

Output

If a valid placement exists, output "`yes`", followed by n distinct integers $h_1, \ldots, h_n$ ($1 \leq h_i \leq n$ for each i), the building heights from left to right.

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

If no valid placement exists, output "`no`".

Sample Input 1	Sample Output 1

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

Sample Input 2	Sample Output 2
`5 3 4`	`no`

Sample Input 3

```
10 3 4
```

Sample Output 3

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

Sample Input 4

```
4 1 4
```

Sample Output 4

```
yes
4 3 2 1
```

Sample Input 5

```
9 1 1
```

Sample Output 5

```
no
```

Sample Input 6

```
6 2 1
```

Sample Output 6

```
yes
5 3 1 2 4 6
```

Problem L: Labour Laws

Time limit: 1 second

Some workplaces use dedicated machines to record work time. In Germany, these are called “Stempeluhren”. CC BY-SA 4.0 by u_h0yvbj97 on Pixabay

The “Arbeitszeitgesetz” (ArbZG) regulates the work and break times of employees in Germany. According to this law, if an employee works more than 6 hours on a given day, the employee **must** take a break of at least half an hour during their shift. If they work for more than 9 hours, their break must be at least 45 minutes. Additionally, working for more than 10 hours on a day is strictly forbidden.

Alexander works at his company’s HR department. As the working hours at his company are very flexible, the employees are allowed to work as many hours or as few as they want (within the legal constraints of course). The monthly salary for each employee thus depends on the time spent working. Therefore, one of Alexander’s monthly tasks is to collect the times each employee has worked on any given day, so that he can calculate and pay out the employees salaries every month.

Unfortunately, today one of his colleagues forgot to record her lunch break and cannot remember how much time she spent on her break. As Alexander does not want to put his colleague at a disadvantage, he wants to use the legally required minimum break time for his calculations.

While calculating the total time his coworker spent at work is easy, correctly calculating the legally required minimum break time can be tricky. Can you help Alexander determine the minimum time his coworker must have spent at lunch today? Note that the time spent at work is the time spent working plus the time spent in break.

Input

The input consists of:

- One line with an integer t ($0 \leq t \leq 1440$), the number of minutes spent at work today.

Output

Output a single non-negative integer, the minimum number of minutes spent in break to make the work time legal.

Sample Input 1	Sample Output 1
495	30

Sample Input 2	Sample Output 2
360	0

Sample Input 3	Sample Output 3
540	30

Sample Input 4	Sample Output 4
0	0

This page is intentionally left (almost) blank.

Problem M: Mex Hex

Time limit: 1 second

The peace and quiet of your home city is being disturbed by the loud purrs and nightly creeping of a nearby magic bobcat. The mayor is sending you to scare it off – you are the city's Guardian for Collateral, Personal, and other Catastrophic Damages after all. You take on the challenge and prepare for scaring away the magnificent animal.

A regular (non-magic) bobcat.
Photo by Becker1999

To defend itself against you, the magic bobcat (as the name suggests) tries to hurt you in unconventional ways. When it attacks, it casts a number of magic *spells*. Each spell is cast with a particular *potential*, which can be expressed as a natural number.[1] If you get hit with spells that have the potentials $p_1, p_2, \dots, p_n$, then the pain that you will feel from them is $\texttt{mex}(\{p_1, p_2, \dots, p_n\})$, where `mex` denotes the Minimum Excluded Value.[2] Note that the spells do not hurt you immediately. Only after all spells are cast will you feel the pain based on the `mex` of their potentials.

To protect yourself, you bring a *shield* which can, also through magic, deflect the spells. Your shield has an *activation duration* d, which means that when you activate the shield, the next d spells do not hit you. Afterwards, the shield must regenerate its magic and cannot be activated for the following d spells. Apart from that, you can activate the shield as often as you want. To ensure that you give the magic bobcat a good scare, the mayor requested that you sneak up to it. As the bobcat would spot the glow of your activated magic shield, the earliest time you can activate the shield is when you stand before the culprit, right before it casts the first spell.

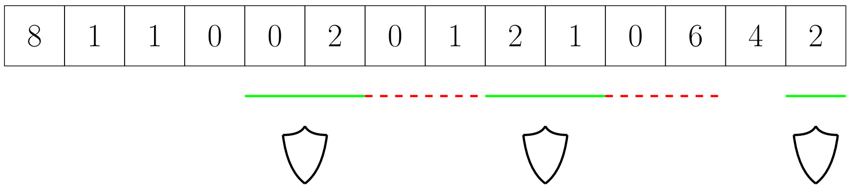


Figure M.1: Illustration of Sample 3. The numbers in the boxes indicate the potentials of the spells. In this example, your shield's activation duration is $d = 2$. By activating the shield right before the 5th, 9th, and 14th spell, you block the spells underlined in green. You cannot activate your shield for the spells that are underlined with a dashed red line, because it is regenerating its magic at these times.

You must now devise a tactic to sustain as little pain as possible. From studying the previous encounters with magic bobcats, you know the potentials of the spells that will be cast against you. What is the minimum pain you can receive from them if you activate your shield optimally?

Input

The input consists of:

- One line with two integers n and d ($1 \leq n \leq 10^5$, $1 \leq d \leq n$), the number of spells and the activation duration of your shield.
- One line with n integers $p_1, \dots, p_n$ ($0 \leq p_i < n$), the potentials of the n spells.

[1] For the purposes of this task, the natural numbers are defined as $\mathbb{N} = \{0, 1, 2, \dots\}$.
[2] Given a set $S \subseteq \mathbb{N}$, $S \neq \mathbb{N}$, $\texttt{mex}(S)$ is the smallest number $m \in \mathbb{N}$ that is *not contained* in S.

Output

Output a single integer, the minimum pain you can receive from the spells.

Sample Input 1

```
5 1
0 1 0 1 0
```

Sample Output 1

```
0
```

You can activate the shield for the first spell, let it regenerate its magic for the second, activate it again for the third, regenerate for the fourth, and activate again on the fifth spell. This means that only the two spells of potential 1 hit you. As $\texttt{mex}(\{1, 1\}) = 0$, you receive 0 pain.

Sample Input 2

```
5 2
0 1 0 1 0
```

Sample Output 2

```
2
```

No matter how you activate your shield, during the time it regenerates, a spell of potential 0 and a spell of potential 1 hit you. Thus, the `mex` of the spells that hit you is at least 2. This amount of pain can be achieved by not using your shield at all.

Sample Input 3

```
14 2
8 1 1 0 0 2 0 1 2 1 0 6 4 2
```

Sample Output 3

```
2
```

Use the shield as depicted in Figure M.1. Then the spells that hit you have potentials 8, 1, 1, 0, 0, 1, 0, 6, and 4. The `mex` of these potentials is 2. It can be shown that there is no way of activating your shield so that the `mex` of the potentials of the remaining spells is 0 or 1.

Sample Input 4

```
4 2
0 1 2 0
```

Sample Output 4

```
1
```

Recall that you cannot activate your shield any earlier than right before the first spell. Thus, you cannot block both spells of potential 0. Instead, by blocking the second and third spell, all spells that hit you have potential 0.