培正喇沙編程挑戰賽
LA SALLE – PUI CHING PROGRAMMING CHALLENGE

第七屆 · 2023

2023 年 8 月 19 日（星期六）

香港培正中學

名稱		時間限制	記憶體限制
A	Always Right	0.5 秒	256 MB
B	Binary Bracket	0.5 秒	256 MB
C	Cross Across the Grid	0.5 秒	256 MB
D	Decisive Duels	2 秒	256 MB
E	Enthusiast of Algorithms	0.5 秒	256 MB
F	Far-reaching Citations	0.5 秒	256 MB
G	GPT Intrusion	0.5 秒	256 MB
H	Handful of Balls	0.5 秒	256 MB
I	Ideal Cutting	0.5 秒	256 MB
J	Joining Two Trees	0.5 秒	256 MB
K	Keep Them Stacked	0.5 秒	256 MB
L	Lift Problem	0.5 秒	256 MB

Pascal	`ppcx64-3.0.0 -O2 -Sg -v0 -dEVAL -XS`
C	`gcc-7 -static -Wno-unused-result -DEVAL -lm -s -O2`
C++	`g++-7 -static -std=c++17 -Wno-unused-result -DEVAL -lm -s -O2`
Python 3	`python3.5 -O -S`

編譯時間限制 10 秒　編譯記憶體限制 512 MB

作者/編製　鄭希哲 招朗軒 鍾懷哲 梁皓寧 龍瑞希 吳有孚 衛家熙
黃正諾 黃敏恆 黃文禮 黃梓駿 黃梓謙 謝凌睿 葉景輝 楊汶璁 袁樂勤

NOTES

1. Output strings are checked case-sensitively.
2. Trailing spaces in output lines are allowed.
3. The end-of-line character at the end of the output is optional.
4. Using scientific notation for real number output is not allowed.
5. 64-bit integer types (e.g. `int64` for Pascal, `long long` for C/C++) may be required for some problems. The C format string token is `%lld`.

A Always Right

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Alice is an expert at solving maze puzzles. One day, her friend Bob gave her a challenge.

Bob: Normal maze puzzles are too easy for you, can you solve a maze without turning left?

Alice: It's simple! Just turn right three times in the same cell!

Bob: What if you must move one step forward before turning right?

Alice doesn't know the solution and asks for your help.

You are given the maze as a $N \times M$ grid. Different cells are represented as follows:

- `#` represents a wall cell.
- `.` represents an empty cell.
- `S` represents the starting cell.
- `E` represents the ending cell.

When the cell in front of you is not a wall cell, you can perform one of the following moves:

1. Move forward one cell.
2. Move forward one cell and turn to your right.

Can you find the minimum number of moves required to reach the ending cell from the starting cell, or determine that it is not possible to do so?

Input

The first line contains two positive integers, N, M, $(4 \leq N, M \leq 800)$.

The following N lines each contain M characters, representing the maze.

The final line contains a single character, `U` (up), `D` (down), `L` (left) or `R` (right), denoting your initial direction.

It is guaranteed that all cells in the outermost layer are wall cells and there are exactly one starting cell and ending cell.

Output

Output a single integer, the minimum number of moves to reach the ending cell, or -1 if the ending cell cannot be reached.

Examples

input

```
6 7
#######
#.....#
#..##.#
#.S...#
#E....#
#######
U
```

output

```
12
```

input

```
5 4
####
#E.#
#S.#
#..#
####
R
```

output

```
5
```

Binary Bracket

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

At the 2023 Wimbledon Championships, world No. 42 Marketa Vondrousova won the Ladies' Singles title, becoming the first ever unseeded competitor to do so. On her way to winning the title, she caused upsets against seeds Veronika Kudermetova, Donna Vekic, Marie Bouzkova, Jessica Pegula, and finally Ons Jabeur in the final.

As a tennis fan, Bob notices the high number of upsets in the Ladies' Singles bracket in the championships this year and wonders if this has anything to do with Vondrousova winning the title. He comes up with the following scenario:

Let there be a single elimination tournament with 2^K competitors, each player having a "power index" P_i. The tournament consists of K rounds, and there are 2^{K-i} matches in the i-th round (The formal proceedings of the tournament is stated below). Each match has one winner (draws do not occur) who moves on to the next round, and the loser is eliminated from the tournament. We define a match as an "upset" if $P_{\text{winner}} < P_{\text{loser}} - X$ where X is a constant given to you.

Formally, the tournament consists of $2^K - 1$ matches. They are held as follows: the players are enumerated from 1 to 2^K, initially in input order, then split into pairs: player number 1 plays against player number 2, player number 3 plays against player number 4 (exactly in this order), and so on (so, 2^{K-1} games are played in this round). When a player loses a game, they are eliminated.

After that, only 2^{K-1} players remain. If only one player remains, they are declared the champion; otherwise, 2^{K-2} games are played in the next round: in the first one of them, the winner of the game "player 1 vs player 2" plays against the winner of the game "player 3 vs player 4", then the winner of the game "player 5 vs player 6" plays against the winner of the game "player 7 vs player 8", and so on.

This process repeats until only one player remains.

Bob now wonders, for each of the 2^K competitors, what is the minimum number of upsets in total (of the $2^K - 1$ matches) for that competitor to become the champion?

Input

The input consists of two lines.

The first line consists of 2 integers K ($1 \le K \le 18$) and X ($0 \le X \le 10^9$).

The second line consists of 2^K integers P_1 to P_{2^K}, representing the power indices of the competitors. ($1 \le P_i \le 10^9$).

Output

Output 2^K integers on a line, one for each player, the minimum number of total upsets in order for them to win the tournament.

Examples

input

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

output

```
2 0 1 0
```

input

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

output

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

C

Cross Across the Grid

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

You are given a $N \times N$ grid where N is always odd.

Each cell in this grid contains an uppercase English letter. The grid has concentric layers, and you can think of each layer as a ring. The outermost layer is layer 1, the next one is layer 2, and so on until the center of the grid.

For each layer, you can rotate its contents (clockwise or anticlockwise) for arbitrary times. Each rotation corresponds to shifting the content of the ring by one cell position.

For example, given the following 3×3 grid, the center would be `X` and there is only one layer, define it as layer 1. Starting from the top-left corner of the layer in the clockwise direction, layer 1 would be `ABCFKPHD`.

```
ABC
DXF
HPK
```

If we rotate layer 1 in the clockwise direction 2 times, the grid will become the following.

```
HDA
PXB
KFC
```

In the *rotated* grid, `HAXKC` are on the diagonals while others do not.

Your task is to determine the minimum number of rotations to make the two diagonals of the grid (forming an X shape) have the same character. It is guaranteed the solution always exists with the given input.

Input

The first line contains a single integer, N, the dimension of the grid ($1 \leq N \leq 100$, N is odd).

The next N lines each contain N characters (without spaces between them), representing the $N \times N$ grid.

Output

Output a single integer, the minimum number of rotations required to make the two diagonals of the grid have the same character in each cell.

Examples

input

```
5
TYEKL
RDEBP
EEEEE
XHEFY
YUEWD
```

output

```
3
```

input

```
9
NMJIITCUS
LXRQWKIXL
UIIKXDIHV
UBTFITYDO
IXKIIILSI
ABCSIPMLJ
YYIFIFIIM
CKINGHZGY
JELGIUBYY
```

output

```
6
```

Note

In the first example, the minimum number of rotations needed will be rotating layer 1 in the anti-clockwise direction 2 times and rotating layer 2 in the clockwise direction 1 time.

D Decisive Duels

Time Limit: 2 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

David is an enthusiastic badminton player who competes in tournaments regularly. There is just one caveat - he's actually quite bad at badminton and, therefore, frequently loses in the first round. Therefore he asks his friend, Dave, for help. Dave is a master in parallel universes, and after intensive research, he found out that all David's matches can be represented using a long binary string S, whereas each match is a substring of the long string. Each character represents a point, `1` means David wins the point, and a `0` means his opponent wins the point. A match progresses by iterating through the characters of the substring in order until one player wins, and the remainder of the string is discarded. The universe will implode if the whole substring is scanned without producing a winner.

Since David lives in the far future, badminton rules have changed slightly. To win, a player must **both**

1. Score at least k points (k can differ across different games and is denoted as k_i for the i-th game)
2. Score at least two more points than their opponent

Since David is still horrible at badminton, he asks Dave to help using his latest timeline-editing tool. Since the tool is still relatively new, it can only be applied on a copy of the substring and does not persist to the other timelines, so **all queries are independent**.

David now has Q queries for you. For each query, given the substring defined by l_i and r_i (which is $S_{l_i}S_{l_i+1}S_{l_i+2}\dots S_{r_i-1}S_{r_i}$) and k_i, what is the minimum number of insertions (adding a character anywhere in the substring) needed such that the universe does not implode and he wins the match?

Input

The first line contains two integers N and Q ($1 \leq N, Q \leq 10^5$), the length of the binary string and the number of queries respectively.

The second line of the input consists of the binary string S of length N.

The i^{th} of the remaining Q lines each contains three integers l_i, r_i, k_i ($1 \leq l_i \leq r_i \leq N$, $0 \leq k_i \leq 10^5$), the index of the leftmost and rightmost character of the substring and minimum points needed to win.

Output

Output Q lines, one for each query:

The minimum number of insertions needed such that the universe does not implode and David wins the match.

Example

input

```
13 3
0110110000100
1 13 7
3 6 3
2 11 7
```

output

```
3
0
2
```

Note

Let's consider the first query in the example. The queried substring is `0110110000100`, with k being 7.

One way to use 3 insertions to change the string into `0110110` `1` `0001` `11` `00`, with the added `1`s underlined.

The match progresses as such:

Character scanned	Current score (David : Opponent)
`0`	$0 : 1$
`1`	$1 : 1$
`1`	$2 : 1$
`0`	$2 : 2$
`1`	$3 : 2$
`1`	$4 : 2$
`0`	$4 : 3$
`1`	$5 : 3$
`0`	$5 : 4$
`0`	$5 : 5$
`0`	$5 : 6$
`1`	$6 : 6$
`1`	$7 : 6$
`1`	$8 : 6$

We note that the `00` at the end of the string is not scanned since the match finishes with David winning before that.

It is also possible to show that 3 insertions are the minimum number needed to ensure David wins.

Enthusiast of Algorithms

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

In this Summer holiday, Bob has participated in the recent Codeforces round and it was his very first online programming contest experience! By ranking the 7156^{th}, Bob successfully got the title Newbie and became a grey contestant in Codeforces. Glancing through the final standing, Bob saw a group of Legendary Grandmasters solving all the problems with incredible speed. He realized that there are still a lot of hard work for him to become one of them in the future.

Motivated by the outstanding results of top programmers, Bob decided to start with learning more algorithms. Yesterday, he borrowed an Algorithm Book in the library. Inside the book, algorithms are categorized into N categories, such as Graph, String, etc. In the i^{th} category, there are a_i different algorithms. There are only K days left in the Summer holiday! Bob is going to learn exactly M algorithms each day in the following K days. For the sake of consistency, the M algorithms learnt in each day should be belonging to the same category.

Bob really wants to become a Legendary Grandmaster, he seeks your help in deciding the number of algorithms to learn each day. Therefore, your job is to find the maximum value M, such that Bob will be learn exactly M different new algorithms under the same category each day in the following K days. Note that it is possible to have some algorithms missed out in Bob's learning process as the time remaining is limited in the Summer holiday.

Input

The first line of the input contains two integers N $(1 \le N \le 10^5)$ and K $(1 \le K \le 10^5)$, denoting the number of categories and the number of days remaining in the Summer holiday respectively.

The second line of the input contains N integers $a_1, a_2, \ldots, a_N$ $(1 \le a_i \le 10^5)$ each separated by a space, where a_i represents the number of different algorithms under category i.

It is guaranteed that $\sum_{i=1}^{N} a_i \ge K$.

Output

The first and the only line of the output should contain a single integer M representing the maximum number of different new algorithms Bob can learn each day in the remaining days of the Summer holiday.

Example

input

```
5 5
4 6 7 3 1
```

output

```
3
```

Note

Bob can learn 3 algorithms under category 1 on day 1. 3 algorithms under category 2 on each of day 2 and 3. 3 algorithms under category 3 on each of day 4 and 5. It can be proven that 3 is the maximum number of algorithms for Bob to learn each day.

F Far-reaching Citations

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Managing citations in academic writing can be challenging. As you are near the completion of a paper draft, it's common to realize that you've lost track of the sources for your ideas. Properly citing relevant papers is essential to highlight their contributions and establish the novelty of your work. Failure to do so not only makes it difficult for reviewers to assess your paper but also raises concerns about plagiarism. Inadequate citations can ultimately diminish your paper's chances of acceptance.

Rebecca needs help assessing her citation needs for her upcoming publication. The world of academic papers is complex, with some being entirely original while others build upon existing works. There are N published papers, labeled as s_i for $1 \le i \le N$. Certain papers directly extend earlier ones by adding new content, written as $s_i = s_j \mid\mid u_i$, where u_i is a non-empty string appended to a previous work s_j. If a paper is created independently, it is represented as $s_i = u_i$.

Rebecca's own paper, represented as string t, is in question. For each distinct pair of indices i and j $(1 \le i \le j \le |t|)$, if the substring $t[i..j] = t_i t_{i+1} .. t_j$ occurs x times within paper s_k, she must add x citations to paper k. For instance, if $t =$ `abc` and $s_1 =$ `abab`, the substring $t[1, 2] =$ `ab` alone would contribute 2 citations to paper s_1. The total count of citations required is the sum across all preceding papers s_k $(1 \le k \le N)$. The task is to determine this total.

Input

The input begins with an integer N $(1 \le N \le 10^5)$, denoting the number of published papers.

Subsequently, there are N lines, the i-th line consisting an integer $j\,(0 \le j < i)$ and a string u_i, separated by a space. A positive j means that paper s_i directly extends an existing work s_j (so $s_i = s_j \mid\mid u_i$), while $j = 0$ indicates that s_i is an independent paper ($s_i = u_i$).

The final line holds the string t $(1 \le |t| \le 10^5)$, representing Rebecca's own paper.

All input strings are composed of lowercase characters. The combined length of all u_i strings is guaranteed not to exceed 10^5.

Output

The output is a single integer representing the total count of citations required for Rebecca's own paper.

Hint: The range of possible outputs sits within the range of `unsigned long long`.

Examples

input

```
3
0 b
1 c
0 dc
bcdc
```

output

```
9
```

input

```
5
0 ab
1 bc
1 bcd
1 cde
0 abc
abcde
```

output

```
39
```

Note

The first sample has 3 papers, $s_1 =$ `b`, $s_2 =$ `bc`, $s_3 =$ `dc`, among which 5 distinct substrings of $t =$ `bcdc` appeared:

- $t[1,1] =$ `b`, which appeared $1+1+0=2$ times in the papers.
- $t[2,2]=t[4,4] =$ `c`, each appeared $0+1+1=2$ times in the papers, so they contribute $2 \times 2 = 4$ citations in total.
- $t[3,3] =$ `d`, which appeared $0+0+1=1$ time in the papers.
- $t[1,2] =$ `bc`, which appeared $0+1+0=1$ time in the papers.
- $t[3,4] =$ `dc`, which appeared $0+0+1=1$ time in the papers.

In total, the number of citations required is $2+4+1+1+1=9$.

GPT Intrusion

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

The La Salle - Pui Ching Programming Challenge is being intruded by GPT!

Many submissions from different contestants are suspected to be written by the GPT model, so it is believed that GPT has hacked into the judge and submitted source codes that are written by itself in different contestants' name. To recover from the damage caused by GPT, all of the submissions need to be checked, and those that are suspected to be written by GPT need to be removed.

Luckily, since the output length of the GPT model is limited, if the source code it writes exceeds 500 characters, only the first 500 characters will be kept (including spaces and end of line characters), and the rest of the output will be replaced by the sentence "As an AI model, my output is limited to 500 characters." followed by an end of line character.

To reduce the number of submissions to be manually checked, as a contestant, please write a program to check if the given source code is suspected to be written by GPT and exceeds its output length.

Input

The first line of the input contains an integer N $(1 \leq N \leq 500)$, the number of lines of the given source code .

The i^{th} of the remaining N lines of the input contains a string S_i $(1 \leq |S_i| \leq 1000)$ of ASCII printable characters, the i^{th} line of the given source code.

It is guaranteed that the sum of $|S_i|$ doesn't exceed 1000.

$|S_i|$ is the length of string S_i (including spaces and the end of line character).

Output

Output `Yes` if the source code given is suspected to be written by GPT and exceeds its output length, otherwise output `No`.

Examples

input

```
6
#include <cstdio>
using namespace std;
int main() {
    printf("No\n");
    return 0;
}
```

output

```
No
```

input

```
15
#include <iostream>
#include <string>
// The main function of the program
int main(int argc, char *argv[]) {
    // Declares an integer n
    int n;
    // Reads an integer from the standard input stream,
    // and store the value into the integer variable n
    std::cin >> n;
    // Declares an array of strings s, with a size of 1000.
    std::string s[1000];
    // A for-loop that loops from 0 to n - 1,
    // looping a total of n times
    for (int i = 0; i < n; i++) {
        // Reads a strAs an AI model, my output is limited to 500 characters.
```

output

```
Yes
```

input

```
16
#include <iostream>
#include <string>
// The main function of the program
int main(int argc, char *argv[]) {
    // Declares an integer n
    int n;
    // Reads an integer from the standard input stream,
    // and store the value into the integer variable n
    std::cin >> n;
    // Declares an array of strings s, with a size of 1000.
    std::string s[1000];
    // A for-loop that loops from 0 to n - 1,
    // looping a total of n times
    for (int i = 0; i < n; i++) {
        // Reads a strAs an AI model,
my output is limited to 500 characters.
```

output

```
No
```

Note

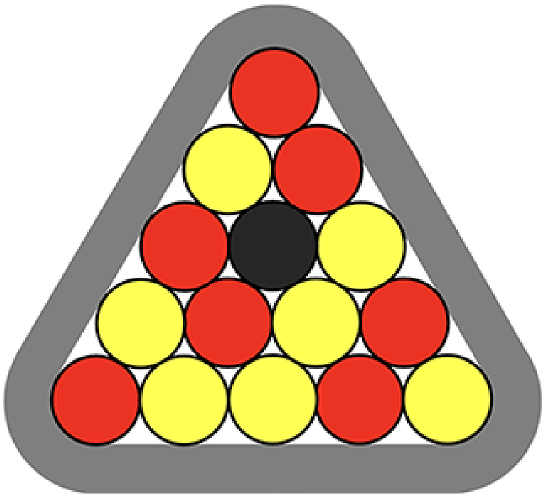
Any deviations from the behaviour stated above will be deemed as human produced code.

H Handful of Balls

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Vincent and Tommy are playing English pool. The game is played with 15 coloured *object balls* (two groups of seven balls – red and yellow – and a black 8-ball) and a white *cue ball*, where they take turns to *pocket* their own set of coloured balls, followed by the 8-ball, by using the pool cue to strike the cue ball to collide with those object balls.

Before each game, they will have to pick the balls from under the table, and with the help of a *rack* set up the 15 object balls on the table in the following specific colour pattern, which is of the shape of an equilateral triangle of side-length 5:



Vincent finds it to be a very natural hand gesture to grab three balls at a time in the form of a triangle of side-length 2 (call it a "*3-ball-triangle*"), with those three balls tightly packed with each other and fit right in the size of his palm. He is also too lazy to rearrange the balls once they have been placed inside the rack. Vincent therefore wonders, if the colours of the balls are to be ignored, whether he can directly fill up the triangular rack with 3-ball-triangle's given the side-length of the rack. Can you help him solve the problem?

Input

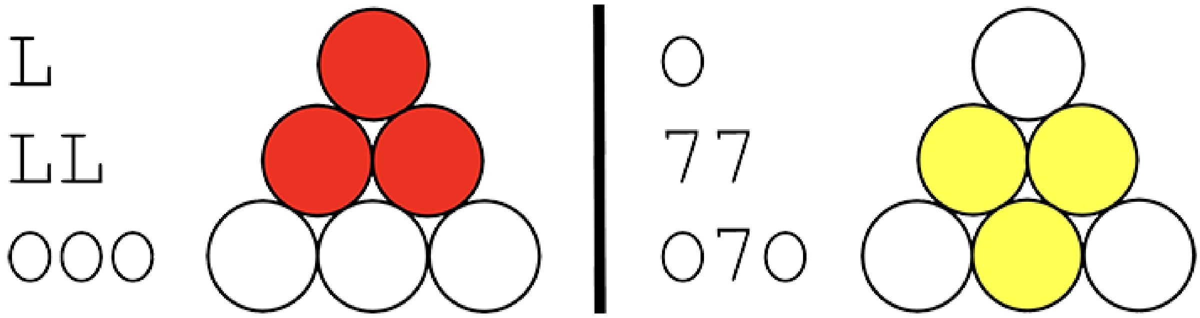
The only line of input contains a single integer N $(1 \leq N \leq 100)$, the length of the side of the triangular rack.

Output

If it is impossible to fill up the triangular rack of side-length N with 3-ball-triangle's, output `Impossible`.

Otherwise, output N lines, with the k-th line containing k characters, that represent the resulting rack to be divided into 3-ball-triangle's. All the outputted characters must be either `L` or `7`.

The rack is to be presented as if all the rows are left-aligned, and should be able to be divided into 'L'-shaped and '7'-shaped regions each with three corresponding characters. Three `L`s form a L-shaped region to represent a 3-ball-triangle pointing upward, and three `7`s form a 7-shaped region to represent a 3-ball-triangle pointing downward. To illustrate this, consider the following (failed) attempts when $N = 3$ (Sample 2): (Note that the `O`s are placeholders that should not appear in your output)



Examples

input	output
2	L LL

input	output
3	Impossible

I Ideal Cutting

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

Bob has a convex polygon with N vertices, and he wants to cut it into $N-2$ triangles using exactly $N-3$ straight line segments, with each starts and ends at two distinct vertices of the polygon. Notice that there must not be two segments intersecting each other, but they can share the same end point.

Bob knows that simply cutting the polygon is an easy job, therefore while cutting it he also wants the triangles look beautiful. Ideally, the triangles are the most beautiful if all of their areas are the same, i.e. the variance of the areas is 0, but Bob knows that it may not be possible. Instead of only seeking for the ideal cutting, he wants to know the minimum variance of areas of the triangles if he cut the polygon optimally, so that they are as beautiful as possible.

Bob asks you for help since he is busy stacking papers from problem K. Please calculate the minimum possible variance of areas of the triangles.

While Bob is stacking the papers, he also reminds you that variance equals the following formula:

$$\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}$$

where μ is the mean of all n numbers x_1, x_2, ..., x_n.

Input

The first line contains one integer N ($3 \le N \le 100$), the number of vertices of the polygon.

The following N lines each contains two integers x and y ($-1000 \le x, y \le 1000$), representing the x-coordinate and y-coordinate of a vertex. It is guaranteed that the vertices form a convex polygon, and they are given in anti-clockwise order.

Output

Output one number, the minimum possible variance of areas of the triangles. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}. Formally, let your answer be x and let the correct answer be y. Your answer is accepted if and only if $\frac{|x-y|}{max(1,|y|)} \le 10^{-6}$.

Examples

input

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

output

```
0.2500000000
```

input

```
4
0 0
4 0
4 2
2 3
```

output

```
0.0000000000
```

J Joining Two Trees

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

As suggested by the task name, we are going to join two trees together!

You are given two trees: T_1 with N_1 vertices and T_2 with N_2 vertices. The vertices of T_1 are numbered from 1 to N_1, while those of T_2 are numbered from $N_1 + 1$ to $N_1 + N_2$. A tree of N vertices is a connected undirected graph with $N - 1$ edges.

Let $dist(x, y)$ be the distance between vertices x and y. The distance between two vertices is the number of edges on the simple path between them.

The diameter of a graph is defined to be the maximum distance between any two vertices. You may have noticed that there may be multiple vertex pairs that have their distance equal to the diameter.

Now, you need to add exactly one edge between a vertex in T_1 and a vertex in T_2 to create a new tree T. Find the maximum number of vertex pairs in T that have their distance equal to the diameter of T.

In other words, maximize the number of unordered pairs (x, y) such that $dist(x, y) = \max_{1 \leq u,v \leq N_1+N_2} dist(u, v)$ in the resulting tree T.

Input

The first line contains two integers N_1 and N_2 ($2 \leq N_1, N_2 \leq 10^5$).

The next $N_1 - 1$ lines each contains two integers x and y, representing an edge connecting vertex x and vertex y in T_1 ($1 \leq x, y \leq N_1$, $x \neq y$).

The next $N_2 - 1$ lines each contains two integers x and y, representing an edge connecting vertex x and vertex y in T_2 ($N_1 + 1 \leq x, y \leq N_1 + N_2$, $x \neq y$).

Output

A single integer, the maximum number of vertex pairs in T such that their distance is equal to the diameter of T.

Examples

input

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

output

```
16
```

input

```
8 3
1 2
2 3
3 4
4 5
5 6
3 7
7 8
9 10
10 11
```

output

```
6
```

Note

In the first sample, adding an edge between vertex 1 and vertex 6 will yield 16 vertex pairs with distances equal to the diameter of T, which is the maximum number of vertex pairs you can get.

In the second sample, the two given trees are shown in the following figure:

Adding an edge between vertex 4 and vertex 10 will yield 6 vertex pairs with distances equal to the diameter of T, which is the maximum number of vertex pairs you can get. The vertex pairs are $(1, 6), (1, 9), (1, 11), (8, 6), (8, 9), (8, 11)$.

K

Keep Them Stacked

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

This year, you joined PLT (Programming Language Tournament) with Alice and Bob, but all of you were very struggled as the problems were tremendously hard.

"We can't even do a single problem..." Alice mumbled.

"Look at that team! They are stacking the problem statement papers for the problems they have solved!" Bob yelled.

You noticed all the problem statement papers were rectangles with different widths and heights. You then said accordingly, "We should stack papers as well! We could pretend we have solved many problems already!" "...But do you know what is the smallest possible area to stack these 3 papers without rotation?" Bob smirked.

Notice that intersected areas of different papers are only counted once. Please find the smallest possible area to stack those 3 papers without rotation if you could place the papers anywhere you want, as you and Bob were interested to figure out the answer.

Input

The first line contains 6 positive integers, $H_1, W_1, H_2, W_2, H_3, W_3$ $(1 \leq H_i, W_i \leq 100)$, where H_i is the height of the i^{th} rectangle paper and W_i is the width of the i^{th} rectangle paper.

Output

Output a single integer, the smallest possible area to stack those 3 papers without rotation.

Example

input	output
1 5 3 2 5 4	21

This page is intentionally left blank.

L Lift Problem

Time Limit: 0.5 seconds
Memory Limit: 256 MB
Input: standard input
Output: standard output

One day, Leo enters a desolate building that has exactly $10N$ floors and N lifts numbered from 1 to N.

In the building, the N lifts are assigned to particular floors so that the time spent waiting for a lift on every floor is shortened. More specifically, there are N evenly separated waiting floors for the lifts to stay on, the i-th of which is floor $10i - 5$. When the lifts are not moving, they always stay on one of the waiting floors. Also, no lifts share the same waiting floor. Initially, lift i is on the i-th waiting floor.

By taking the lifts, Leo can only move between floor x to floor y, where both x and y are not waiting floors. Each time he wants to move from floor x to floor y, the lift on the closest waiting floor will go to floor x. **In the case of ties, the lift on the lower floor will go there.** Then, the lift will take him to floor y. After the lift reaches floor y, every lift will immediately return to one of the waiting floors according to its current position. More precisely, a lift on a lower floor will move to a lower waiting floor.

Currently, Leo is on floor F (It is guaranteed that F is not a waiting floor). When he is walking around floor F, he suddenly comes up with a permutation P with N distinct integers from 1 to N. Therefore, he decides to shuffle the lifts into the permutation P so that lift P_i is on the i-th waiting floor. As there are no escalators and stairs in the building, he can only shuffle the lifts by taking the lifts to move between floors.

As Leo needs to leave the building after an hour, he needs to shuffle the lifts into the permutation P within $5N$ lift rides. Leo thinks it is too difficult and calls for your help. Can you teach Leo how he should move to complete it?

Note that you are not required to minimize the number of lift rides Leo takes.

Input

The first line contains two integers, N and F, the number of lifts in the building and the floor Leo currently on respectively. ($3 \le N \le 200$, $1 \le F \le 10N$)

The second line contains a permutation P of N integers, the i-th integer denotes the number of the lift on the i-th waiting floor after the moves.

It is guaranteed that F is not a waiting floor.

Output

On the first line, output one integer K ($0 \le K \le 5N$), the number of lift rides Leo takes.

On the second line, output K integers, the i-th of which represents the floor Leo moves to in the i-th move.

It can be proven that it is always possible to do so. If there are multiple answers, you can output any of them.

Examples

input

```
3 13
3 1 2
```

output

```
2
23 2
```

input

```
5 27
3 1 5 4 2
```

output

```
7
2 13 24 48 37 26 38
```

Note

In the first sample test, Leo took two lift rides.

1. Leo moved from floor 13 to floor 23 by taking lift 2. After the lift had reached floor 23, it returned to the 2-nd waiting floor, while the other two lifts remained on their initial waiting floor.
2. Leo moved from floor 23 to floor 2 by taking lift 3. After the lift had reached floor 2, it moved to the 1-st waiting floor, while lift 1 and lift 2 moved to the 2-nd and 3-rd waiting floor respectively.