## A. C+=

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Leo has developed a new programming language C+=. In C+=, integer variables can only be changed with a "+=" operation that adds the right-hand side value to the left-hand side variable. For example, performing "`a += b`" when `a` $= 2$, `b` $= 3$ changes the value of `a` to $5$ (the value of `b` does not change).

In a prototype program Leo has two integer variables `a` and `b`, initialized with some positive values. He can perform any number of operations "`a += b`" or "`b += a`". Leo wants to test handling large integers, so he wants to make the value of either `a` or `b` **strictly greater** than a given value $n$. What is the smallest number of operations he has to perform?

### Input

The first line contains a single integer $T$ $(1 \leq T \leq 100)$ — the number of test cases.

Each of the following $T$ lines describes a single test case, and contains three integers $a, b, n$ $(1 \leq a, b \leq n \leq 10^9)$ — initial values of `a` and `b`, and the value one of the variables has to exceed, respectively.

### Output

For each test case print a single integer — the smallest number of operations needed. Separate answers with line breaks.

### Example

**input**

```
2
1 2 3
5 4 100
```

**output**

```
2
7
```

### Note

In the first case we cannot make a variable exceed $3$ in one operation. One way of achieving this in two operations is to perform "`b += a`" twice.

## B. Codeforces Subsequences

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Karl likes Codeforces and subsequences. He wants to find a string of lowercase English letters that contains at least $k$ subsequences `codeforces`. Out of all possible strings, Karl wants to find a shortest one.

Formally, a `codeforces` subsequence of a string $s$ is a subset of ten characters of $s$ that read `codeforces` from left to right. For example, `codeforces` contains `codeforces` a single time, while `codeforcesisawesome` contains `codeforces` four times: **`codeforce`**`s`**`isawe`**`some`, **`codeforce`**`sis`**`awe`**`some`, **`codeforce`**`sis`**`awes`**`ome`, **`codeforce`**`sisaw`**`es`**`ome`.

Help Karl find any shortest string that contains at least $k$ `codeforces` subsequences.

### Input

The only line contains a single integer $k$ $(1 \leq k \leq 10^{16})$.

### Output

Print a shortest string of lowercase English letters that contains at least $k$ `codeforces` subsequences. If there are several such strings, print any of them.

### Examples

**input**

```
1
```

**output**

```
codeforces
```

**input**

```
3
```

**output**

```
codeforcesss
```

## C. Even Picture

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Leo Jr. draws pictures in his notebook with checkered sheets (that is, each sheet has a regular square grid printed on it). We can assume that the sheets are infinitely large in any direction.

To draw a picture, Leo Jr. colors some of the cells on a sheet gray. He considers the resulting picture *beautiful* if the following conditions are satisfied:

- The picture is **connected**, that is, it is possible to get from any gray cell to any other by following a chain of gray cells, with each pair of adjacent cells in the path being neighbours (that is, sharing a side).
- Each gray cell has an **even number of gray neighbours**.
- There are **exactly $n$ gray cells with all gray neighbours**. The number of other gray cells can be arbitrary *(but reasonable, so that they can all be listed)*.

Leo Jr. is now struggling to draw a beautiful picture with a particular choice of $n$. Help him, and provide any example of a beautiful picture.

To output cell coordinates in your answer, assume that the sheet is provided with a Cartesian coordinate system such that one of the cells is chosen to be the origin $(0, 0)$, axes $0x$ and $0y$ are orthogonal and parallel to grid lines, and a unit step along any axis in any direction takes you to a neighbouring cell.

### Input

The only line contains a single integer $n$ ($1 \leq n \leq 500$) — the number of gray cells with all gray neighbours in a beautiful picture.

### Output

In the first line, print a single integer $k$ — the number of gray cells in your picture. For technical reasons, $k$ should not exceed $5 \cdot 10^5$.

Each of the following $k$ lines should contain two integers — coordinates of a gray cell in your picture. All listed cells should be distinct, and the picture should satisdfy all the properties listed above. All coordinates should not exceed $10^9$ by absolute value.

One can show that there exists an answer satisfying all requirements with a small enough $k$.

#### Example

**input**

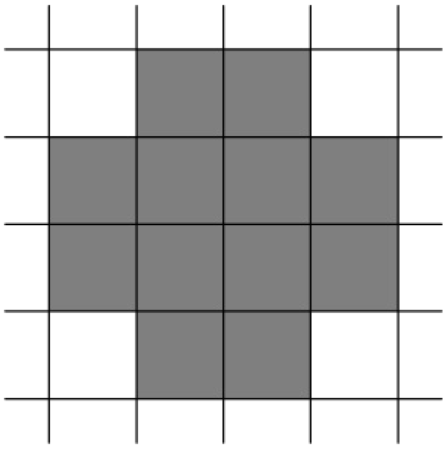
```
4
```

**output**

```
12
1 0
2 0
0 1
1 1
2 1
3 1
0 2
1 2
2 2
3 2
1 3
2 3
```

### Note

The answer for the sample is pictured below:



## D. AND, OR and square sum

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Gottfried learned about binary number representation. He then came up with this task and presented it to you.

You are given a collection of $n$ non-negative integers $a_1, \ldots, a_n$. You are allowed to perform the following operation: choose two distinct indices $1 \leq i, j \leq n$. If before the operation $a_i = x$, $a_j = y$, then after the operation $a_i = x$ AND $y$, $a_j = x$ OR $y$, where AND and OR are bitwise AND and OR respectively (refer to the Notes section for formal description). The operation may be performed any number of times (possibly zero).

After all operations are done, compute $\sum_{i=1}^{n} a_i^2$ — the sum of squares of all $a_i$. What is the largest sum of squares you can achieve?

### Input

The first line contains a single integer $n$ ($1 \leq n \leq 2 \cdot 10^5$).

The second line contains $n$ integers $a_1, \ldots, a_n$ ($0 \leq a_i < 2^{20}$).

### Output

Print a single integer — the largest possible sum of squares that can be achieved after several (possibly zero) operations.

### Examples

**input**
```
1
123
```
**output**
```
15129
```

**input**
```
3
1 3 5
```
**output**
```
51
```

**input**
```
2
349525 699050
```
**output**
```
1099509530625
```

### Note

In the first sample no operation can be made, thus the answer is $123^2$.

In the second sample we can obtain the collection $1, 1, 7$, and $1^2 + 1^2 + 7^2 = 51$.

If $x$ and $y$ are represented in binary with equal number of bits (possibly with leading zeros), then each bit of $x$ AND $y$ is set to $1$ if and only if both corresponding bits of $x$ and $y$ are set to $1$. Similarly, each bit of $x$ OR $y$ is set to $1$ if and only if at least one of the corresponding bits of $x$ and $y$ is set to $1$. For example, $x = 3$ and $y = 5$ are represented as $011_2$ and $101_2$ (highest bit first). Then, $x$ AND $y = 001_2 = 1$, and $x$ OR $y = 111_2 = 7$.

## E. Ski Accidents

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Arthur owns a ski resort on a mountain. There are $n$ landing spots on the mountain numbered from $1$ to $n$ from the top to the foot of the mountain. The spots are connected with one-directional ski tracks. All tracks go towards the foot of the mountain, so there are **no directed cycles** formed by the tracks. There are **at most two tracks leaving each spot**, but many tracks may enter the same spot.

A skier can start skiing from one spot and stop in another spot if there is a sequence of tracks that lead from the starting spot and end in the ending spot. Unfortunately, recently there were many accidents, because the structure of the resort allows a skier to go through dangerous paths, by reaching high speed and endangering himself and the other customers. Here, a path is called dangerous, if it consists of at least two tracks.

Arthur wants to secure his customers by closing some of the spots in a way that there are no dangerous paths in the resort. When a spot is closed, all tracks entering and leaving that spot become unusable.

Formally, after closing some of the spots, **there should not be a path that consists of two or more tracks**.

Arthur doesn't want to close too many spots. He will be happy to find any way to close **at most** $\frac{4}{7}n$ **spots** so that the remaining part is safe. Help him find any suitable way to do so.

## Input

The first line contains a single positive integer $T$ — the number of test cases. $T$ test case description follows.

The first line of each description contains two integers $n$ and $m$ ($1 \leq n \leq 2 \cdot 10^5$) — the number of landing spots and tracks respectively.

The following $m$ lines describe the tracks. Each of these lines contains two integers $x$ and $y$ ($1 \leq x < y \leq n$) — indices of the starting and finishing spots for the respective track. It is guaranteed that at most two tracks start at each spot. There may be tracks in which starting and finishing spots both coincide.

It is guaranteed that the sum of $n$ over all test cases does not exceed $2 \cdot 10^5$.

## Output

For each test case, print a single integer $k$ ($0 \leq k \leq \frac{4}{7}n$) — the number of spots to be closed. In the next line, print $k$ distinct integers — indices of all spots to be closed, in any order.

If there are several answers, you may output any of them. Note that you **don't have to minimize** $k$. It can be shown that a suitable answer always exists.

### Example

**input**

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

**output**

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

## Note

In the first sample case, closing any two spots is suitable.

In the second sample case, closing only the spot $1$ is also suitable.

# F. Lamps on a Circle

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

*This is an interactive problem.*

John and his imaginary friend play a game. There are $n$ lamps arranged in a circle. Lamps are numbered $1$ through $n$ in clockwise order, that is, lamps $i$ and $i+1$ are adjacent for any $i = 1, \ldots, n-1$, and also lamps $n$ and $1$ are adjacent. Initially all lamps are turned off.

John and his friend take turns, with John moving first. On his turn John can choose to terminate the game, or to make a move. To make a move, John can choose any positive number $k$ and turn any $k$ lamps of his choosing on. In response to this move, John's friend will choose $k$ **consecutive** lamps and turn all of them off (the lamps in the range that were off before this move stay off). Note that the value of $k$ is the same as John's number on his last move. For example, if $n = 5$ and John have just turned three lamps on, John's friend may choose to turn off lamps $1, 2, 3$, or $2, 3, 4$, or $3, 4, 5$, or $4, 5, 1$, or $5, 1, 2$.

After this, John may choose to terminate or move again, and so on. However, John can not make more than $10^4$ moves.

John wants to maximize the number of lamps turned on at the end of the game, while his friend wants to minimize this number. Your task is to provide a strategy for John to achieve optimal result. Your program will play interactively for John against the jury's interactor program playing for John's friend.

Suppose there are $n$ lamps in the game. Let $R(n)$ be the number of turned on lamps at the end of the game if both players act optimally. Your program has to terminate the game with at least $R(n)$ turned on lamps within $10^4$ moves. Refer to Interaction section below for interaction details.

For technical reasons **hacks for this problem are disabled.**

## Interaction

Initially your program will be fed a single integer $n$ ($1 \le n \le 1000$) — the number of lamps in the game. Then the interactor will wait for your actions.

To make a move, print a line starting with an integer $k$ ($1 \le k \le n$), followed by $k$ **distinct** integers $l_1, \ldots, l_k$ ($1 \le l_i \le n$) — indices of lamps you want to turn on. The indices may be printed in any order. It is allowed to try to turn on a lamp that is already on (although this will have no effect).

If your move was invalid for any reason, or if you have made more than $10^4$ moves, the interactor will reply with a line containing a single integer $-1$. Otherwise, the reply will be a line containing a single integer $x$ ($1 \le x \le n$), meaning that the response was to turn off $k$ consecutive lamps starting from $x$ in clockwise order.

To terminate the game instead of making a move, print a line containing a single integer $0$. The test will be passed if at this point there are at least $R(n)$ lamps turned on (note that neither $R(n)$, nor the verdict received are not communicated to your program in any way). This action does not count towards the number of moves (that is, it is legal to terminate the game after exactly $10^4$ moves).

To receive the correct verdict, your program should terminate immediately after printing $0$, or after receiving $-1$ as a response.

Don't forget to flush your output after every action.

**Examples**

**input**
```
3
```
**output**
```

0
```

**input**
```
4

1
```
**output**
```

2 1 3

0
```

## Note

When $n = 3$, any John's move can be reversed, thus $R(3) = 0$, and terminating the game immediately is correct.

$R(4) = 1$, and one strategy to achieve this result is shown in the second sample case.

Blank lines in sample interactions are for clarity and should not be printed.

## G. Shifting Dominoes

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Bill likes to play with dominoes. He took an $n \times m$ board divided into equal square cells, and covered it with dominoes. Each domino covers two adjacent cells of the board either horizontally or vertically, and each cell is covered exactly once with a half of one domino (that is, there are no uncovered cells, and no two dominoes cover the same cell twice).

After that Bill decided to play with the covered board and share some photos of it on social media. First, he removes exactly one domino from the board, freeing two of the cells. Next, he moves dominoes around. A domino can only be moved **along the line parallel to its longer side**. A move in the chosen direction is possible if the next cell in this direction is currently free. Bill doesn't want to lose track of what the original tiling looks like, so he makes sure that at any point each domino **shares at least one cell with its original position**.

After removing a domino and making several (possibly, zero) moves Bill takes a photo of the board and posts it. However, with the amount of filters Bill is using, domino borders are not visible, so **only the two free cells of the board** can be identified. When the photo is posted, Bill reverts the board to its original state and starts the process again.

Bill wants to post as many photos as possible, but he will not post any photo twice. How many distinct photos can he take? Recall that photos are different if the pairs of free cells in the photos are different.

## Input

The first line contains two positive integers $n$ and $m$ ($nm \leq 2 \cdot 10^5$) — height and width of the board respectively.

The next $n$ lines describe the tiling of the board, row by row from top to bottom. Each of these lines contains $m$ characters, describing the cells in the corresponding row left to right. Each character is one of U, D, L, or R, meaning that the cell is covered with a top, bottom, left, or right half of a domino respectively.

It is guaranteed that the described tiling is valid, that is, each half-domino has a counterpart in the relevant location. In particular, since tiling is possible, the number of cells in the board is even.

## Output

Print a single integer — the number of distinct photos Bill can take.

### Examples

**input**

```
2 4
UUUU
DDDD
```

**output**

```
4
```

**input**

```
2 3
ULR
DLR
```

**output**

```
6
```

**input**

```
6 6
ULRUUU
DUUDDD
UDDLRU
DLRLRD
ULRULR
DLRDLR
```

**output**

```
133
```

## Note

In the first sample case, no moves are possible after removing any domino, thus there are four distinct photos.

In the second sample case, four photos are possible after removing the leftmost domino by independently moving/not moving the remaining two dominoes. Two more different photos are obtained by removing one of the dominoes on the right.

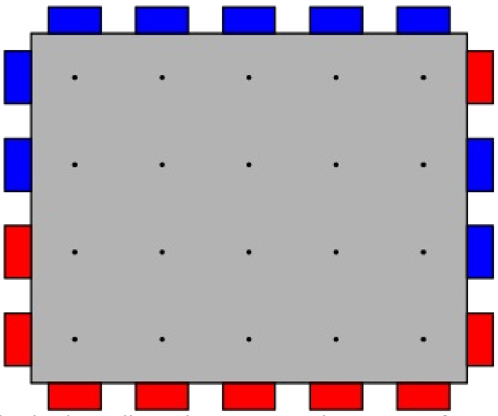
## H1. Breadboard Capacity (easy version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

*This is an easier version of the problem H without modification queries.*

Lester and Delbert work at an electronics company. They are currently working on a microchip component serving to connect two independent parts of a large supercomputer.

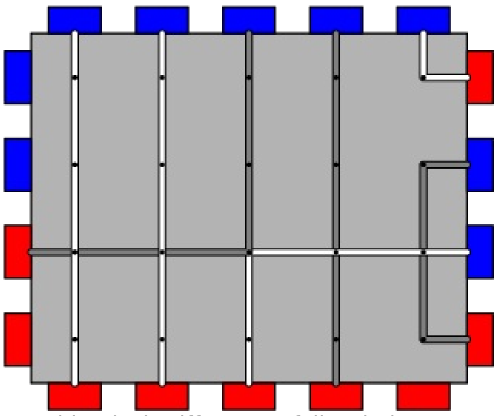
The component is built on top of a *breadboard* — a grid-like base for a microchip. The breadboard has $n$ rows and $m$ columns, and each row-column intersection contains a node. Also, on each side of the breadboard there are ports that can be attached to adjacent nodes. Left and right side have $n$ ports each, and top and bottom side have $m$ ports each. Each of the ports is connected on the outside to one of the parts bridged by the breadboard, and is colored red or blue respectively.



Ports can be connected by wires going inside the breadboard. However, there are a few rules to follow:

- Each wire should connect a red port with a blue port, and each port should be connected to at most one wire.
- Each part of the wire should be horizontal or vertical, and turns are only possible at one of the nodes.
- To avoid interference, wires can not have common parts of non-zero length (but may have common nodes). Also, a wire can not cover the same segment of non-zero length twice.

The *capacity* of the breadboard is the largest number of red-blue wire connections that can be made subject to the rules above. For example, the breadboard above has capacity $7$, and one way to make seven connections is pictured below.



*Up to this point statements of both versions are identical. Differences follow below.*

Given the current breadboard configuration, help Lester and Delbert find its capacity efficiently.

### Input

The first line contains three integers $n, m, q$ $(1 \leq n, m \leq 10^5$, $\boldsymbol{q = 0})$. $n$ and $m$ are the number of rows and columns of the breadboard respectively. *In this version* $q$ *is always zero, and is only present for consistency with the harder version.*

The next four lines describe initial coloring of the ports. Each character in these lines is either R or B, depending on the coloring of the respective port. The first two of these lines contain $n$ characters each, and describe ports on the left and right sides respectively from top to bottom. The last two lines contain $m$ characters each, and describe ports on the top and bottom sides respectively from left to right.

### Output

Print a single integer — the given breadboard capacity.

**Example**

**input**

```
4 5 0
BBRR
RBBR
BBBBB
RRRRR
```

**output**

```
7
```

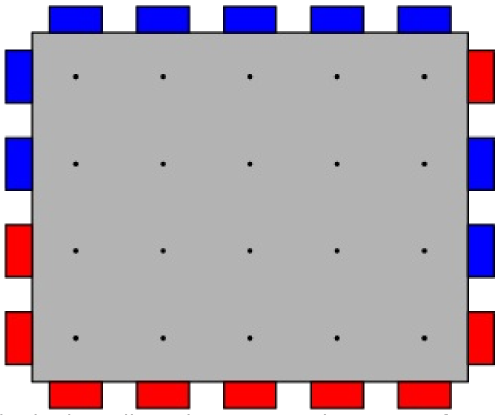
## H2. Breadboard Capacity (hard version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

*This is a harder version of the problem H with modification queries.*

Lester and Delbert work at an electronics company. They are currently working on a microchip component serving to connect two independent parts of a large supercomputer.

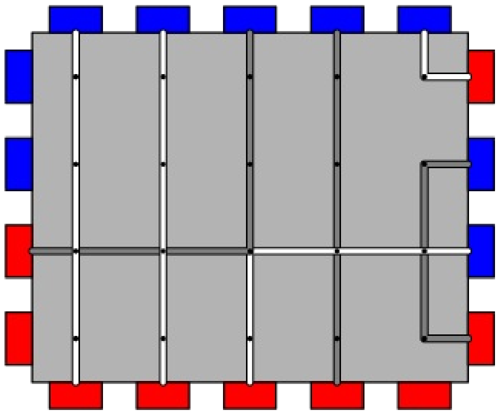
The component is built on top of a *breadboard* — a grid-like base for a microchip. The breadboard has $n$ rows and $m$ columns, and each row-column intersection contains a node. Also, on each side of the breadboard there are ports that can be attached to adjacent nodes. Left and right side have $n$ ports each, and top and bottom side have $m$ ports each. Each of the ports is connected on the outside to one of the parts bridged by the breadboard, and is colored red or blue respectively.



Ports can be connected by wires going inside the breadboard. However, there are a few rules to follow:

- Each wire should connect a red port with a blue port, and each port should be connected to at most one wire.
- Each part of the wire should be horizontal or vertical, and turns are only possible at one of the nodes.
- To avoid interference, wires can not have common parts of non-zero length (but may have common nodes). Also, a wire can not cover the same segment of non-zero length twice.

The *capacity* of the breadboard is the largest number of red-blue wire connections that can be made subject to the rules above. For example, the breadboard above has capacity $7$, and one way to make seven connections is pictured below.



*Up to this point statements of both versions are identical. Differences follow below.*

As is common, specifications of the project change a lot during development, so coloring of the ports is not yet fixed. There are $q$ modifications to process, each of them has the form of "colors of all ports in a contiguous range along one of the sides are switched (red become blue, and blue become red)". All modifications are persistent, that is, the previous modifications are not undone before the next one is made.

To estimate how bad the changes are, Lester and Delbert need to find the breadboard capacity after each change. Help them do this efficiently.

## Input

The first line contains three integers $n, m, q$ ($1 \leq n, m \leq 10^5$, $0 \leq q \leq 10^5$) — the number of rows and columns of the breadboard, and the number of modifications respectively.

The next four lines describe initial coloring of the ports. Each character in these lines is either R or B, depending on the coloring of the respective port. The first two of these lines contain $n$ characters each, and describe ports on the left and right sides respectively from top to bottom. The last two lines contain $m$ characters each, and describe ports on the top and bottom sides respectively from left to right.

The next $q$ lines describe modifications. Each of these lines contains a character $s$, followed by two integers $l$ and $r$. If $s$ is L or R, the modification is concerned with ports on the left/right side respectively, $l$ and $r$ satisfy $1 \leq l \leq r \leq n$, and ports in rows between $l$ and $r$ (inclusive) on the side switch colors. Similarly, if $s$ is U or D, then $1 \leq l \leq r \leq m$, and ports in columns between $l$ and $r$ (inclusive) on the top/bottom side respectively switch colors.

## Output

Print $q + 1$ integers, one per line — the breadboard capacity after $0, \ldots, q$ modifications have been made to the initial coloring.

**Example**

**input**

```
4 5 4
BBRR
```

```
RBBR
BBBBB
RRRRR
L 2 3
R 3 4
U 1 5
D 1 5
```

**output**

```
7
7
9
4
9
```