Division 1

Hosted by:

College of Charleston

Florida International University

Kennesaw State University

University of West Florida

Abstract Art

Peter painted a peculiar piece of art involving a grid of cells. He's not quite happy with it. Some of the colors in his painting are touching, and he doesn't like that. He would like to use a dab of whiteout to separate any different colors that are touching.

The painting consists of an ***r*×*c*** grid of cells where each cell is painted some color. Two colored cells are touching if they share a border edge. Cells sharing only a corner are not considered touching.

Cells are labelled with lowercase letters to denote colors. Peter can replace some of the cells with '**#**' representing whiteout. For example:

```
aaab    aa#b
aabb    a#bb
aaab    aa#b
cccc    ##c#
```

Peter has separated all of the colors using only 6 dabs of whiteout. In order to save whiteout, Peter would like to minimize the number of cells he must dab.

But Peter, in the process of trying to save his painting, may accidentally paint over all of his favorite color! So while using the minimum number of dabs of whiteout, he would like to know which colors could possibly be retained in the painting.

For example the painting:

```
bab
bbb
```

Can be fixed using one dab:

```
b#b
bbb
```

It is possible to save the color ***b*** with a single dab, but it is not possible to use only a single dab and retain the color **a**.

Determine this minimum number of dabs and which colors are possible to save.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with two space-separated integers ***r*** and ***c*** ($1 \le r,c \le 100$), where ***r*** is the number of rows in the painting, and ***c*** is the number of columns.

Each of the next ***r*** lines will contain a string of length exactly ***c***, consisting of only lower case letters, or the symbol '**#**' representing a cell that has already been whited out.

Output

Output exactly two lines. On the first line output a single integer, which is the minimum number of *additional* cells that must be whited out, not counting those that are already whited out. On the second line output a string of lower case letters in increasing alphabetic order. These should be the colors in the painting that can be retained by whiting out a minimum number of cells.

Sample Input	**Sample Output**
2 2 ab ba	2 ab
2 3 aba aaa	1 a
3 4 #a## bcde ##f#	2 abef

In the first example, **a** can be retained with 2 dabs:

a#
#a

And **b** can be retained with 2 dabs:

#b
b#

So, both **a** and **b** can be retained with 2 dabs, just not at the same time.

Count the Bits

Given a value ***k*** and a number of bits ***b***, calculate the total number of 1-bits in the binary representations of all multiples of ***k*** that are between 0 and 2^b-1 (inclusive).

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will consist of a single line containing two space-separated integers ***k*** ($1 \le k \le 1{,}000$) and ***b*** ($1 \le b \le 128$), where ***k*** and ***b*** are as described above.

Output

Output a single integer, which is the total number of 1-bits in the binary representations of all multiples of ***k*** that are between 0 and 2^b-1 (inclusive). Since this number may be very large, output it modulo 10^9+9.

Sample Input	Sample Output
1 4	32
10 5	8
100 7	3
3 28	252698795
11 128	856188165
1 26	872415232
876 128	530649653

Consider the second sample: ***k***=10 and ***b***=5.

2^5-1 = 31. All the multiples of 10 between 0 and 31 are: 10, 20 and 30.

10 = 01010b (2 1-bits)
20 = 10100b (2 1-bits)
30 = 11110b (4 1-bits)

That's a total of 2+2+4=8 1-bits.

Exam

You and your friend have just taken a True/False exam. Your friend has been to see the instructor, so they know how many answers they got right (but not which ones). You compare notes: you know your answers and your friend's answers. What is the maximum number of answers you could have gotten right?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line containing a single integer n ($0 \le n \le 1{,}000$), which is the number of answers your friend got right on the exam.

Each of the next two lines will contain a string s ($\max[n,1] \le |s| \le 1{,}000$, $s \in \{\mathbf{T},\mathbf{F}\}^*$). The two strings will be of the same length. The first line represents your answers; the second line represents your friend's answers. The order of answers is the same in both strings: the first letter is the answer to question 1, the second to question 2, and so on.

Output

Output a single integer, which is the maximum number of answers you could have gotten right.

Sample Input	Sample Output
3 FTFFF TFTTT	2
6 TTFTFFTFTF TTTTFFTTTT	9

Illiteracy

Illiteracy is a simple puzzle game. After the contest, if you'd like to play it, you can find it here: **https://le-slo.itch.io/illiteracy**. Of course, during the contest, it won't be accessible (and you've got better things to do!)

The game has a string of 8 icons. The icons in the game are very artistic, but for simplicity, we'll just call them **A**..**F**. Clicking any icon has a unique effect on the other icons. Most of the icons *Rotate* other icons. That means that they change **A**→**B**, **B**→**C**, **C**→**D**, **D**→**E**, **E**→**F**, and **F**→**A**.

There are 8 icon positions in a row, numbered left to right, 1 to 8. Here's what each of the icons do when clicked:

- **A**: *Rotates* the icon immediately to the left, if there is one, and immediately to the right, if there is one.
- **B**: If not on the end, changes the icon immediately to the right to be same as the one immediately to the left (does nothing on the ends). This is the only icon that doesn't *Rotate* other icons.
- **C**: *Rotates* the mirror image (when clicked in position ***x***, *Rotates* 9-***x***. e.g. clicking 1 *Rotates* 8, 2 *Rotates* 7, etc.)
- **D**: *Rotates* all of the icons between this one and the closest end. (e.g. clicking 3 *Rotates* 1 and 2, 5 *Rotates* 6, 7 and 8. Clicking this icon on the end does nothing.)
- **E**: *Rotates* the closest end, and also the position which is the same distance in the opposite direction. (e.g. clicking 1 does nothing, 2 *Rotates* 1 and 3, 3 *Rotates* 1 and 5, 5 *Rotates* 8 and 2, 7 *Rotates* 8 and 6, etc.)
- **F**: *Rotates* another position with this pattern: Clicking 1 *Rotates* 5, 2 *Rotates* 1, 3 *Rotates* 6, 4 *Rotates* 2, 5 *Rotates* 7, 6 *Rotates* 3, 7 *Rotates* 8, and 8 *Rotates* 4. In other words, clicking an icon in an odd position *Rotates* (***x***+9)/2, even *Rotates* ***x***/2.

Given a starting and target configuration, what's the smallest number of steps needed to get from the start to the target?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

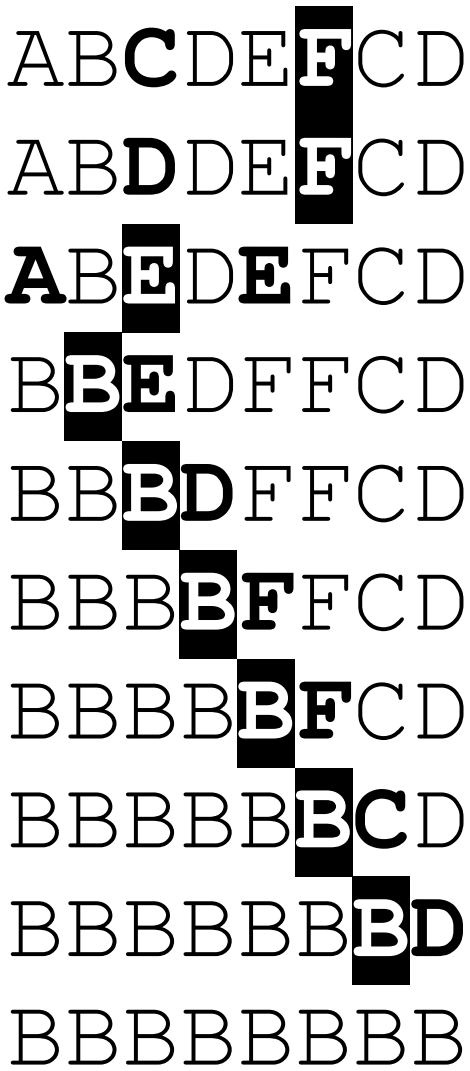
Each test case will consist of exactly two lines. Each line will have a string of length exactly 8, consisting only of the upper-case letters **A**, **B**, **C**, **D**, **E** and/or **F**. The first line holds the starting position, the second holds the target.

Output

Output a single integer, which is the smallest number of steps needed to get from the start to the target, or -1 if it isn't possible.

Sample Input	Sample Output
ABCDEFCD BBBBBBBB	9
BBBBBBBB ABCDEFCD	-1

Here is one possible way to solve the first case in the minimum 9 steps. The icon clicked is in **Inverse**, the icons which change are in **Bold**.



Inversions

Consider a sequence of ***n*** integers, all of them between 1 and ***k*** (inclusive). Some of the integers are missing, and are replaced with 0s.

An *inversion* is a pair of values $\mathbf{a}_i$ and $\mathbf{a}_j$ in the sequence, where $\mathbf{i}<\mathbf{j}$, but $\mathbf{a}_i>\mathbf{a}_j$. What's the maximum number of inversions possible if the missing integers are all between 1 and ***k*** inclusive?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will start with a line with two space-separated integers ***n*** ($1 \le n \le 200{,}000$) and ***k*** ($1 \le k \le 100$), where ***n*** is the length of the sequence and ***k*** is the maximum value of elements of the sequence.

Each of the next ***n*** lines will contain a single integer ***x*** ($0 \le x \le k$). This is the sequence, in order, with 0s representing the missing values.

Output

Output a single integer, which is the maximum number of inversions possible.

Sample Input	Sample Output
6 9 0 8 4 3 0 0	15
10 9 5 2 9 0 7 4 8 7 0 0	28
10 9 7 4 0 0 8 5 0 0 3 1	36

In the first example, if you replace the 0s like this:

9 8 4 3 2 1

Then every pair of numbers in the sequence is an *inversion*, for a total of 15.

Knockout

The solitaire game *Knockout* is played as follows. The digits from 1 to 9 are written down in ascending order. In each turn, you throw a pair of six-sided dice; you sum the dice, and cross out some set of digits, of your choice, that sum to the same total. If you cannot, the game ends and your score is the remaining digits, taken as a single number. Otherwise, you throw the dice again and continue.

This game can be played to either minimize or maximize your score. Given a position of the game (what digits remain) and a roll of the dice, determine which digits you should remove and what your expected score would be for both versions of the game, assuming you make the best moves possible for whichever version you're playing for the remainder of the game. The expected score is the sum of all possible scores weighted by their probabilities (presuming optimum play).

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will consist of a single line containing a string of digits ***d*** ($1 \le |\boldsymbol{d}| \le 9$) and two integers ***a*** and ***b*** ($1 \le \boldsymbol{a},\boldsymbol{b} \le 6$), all separated by spaces. The string of digits ***d*** will contain a subset of the digits 1..9 in ascending order, with no digit appearing more than once. This is the current state of the game. The integers ***a*** and ***b*** represent your current throw of the dice.

Output

Output two lines, each with two parts. First, output the digits that you eliminate with your throw of the dice, as a string of digits in ascending order. If you cannot eliminate any digits, output -1. Then, output the expected score as a real number rounded to five decimal places. Output a space between the parts.

The first line represents the best result when minimizing your score and the second line represents the best result when maximizing your score. Note that it is impossible for two different combinations of digits to yield the same expected score.

Sample Input	Sample Output
1345 1 1	-1 1345.00000 -1 1345.00000
12349 3 1	13 151.70370 4 401.24546

Troop Mobilization

You are playing a strategy games in which you are required to mobilize an army. The army consists of different types of troops, each of which has a cost, health, and potency. You can acquire any combination of the troop types, even fractional, such that the total cost is no more than the amount of money you have to spend. The strength of the army is equal to its *total* health value multiplied by its *total* potency (i.e. $(\Sigma a_i h_i) \times (\Sigma a_i p_i)$, where h_i is the health of troop type **i**, p_i is the potency of troop type **i**, and a_i is the amount acquired of troop type **i**). What is the greatest strength you can achieve given the troops available and the money in your coffers? You may assume that there will always be sufficient troops to buy as many as you can afford.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with two space-separated integers n ($1 \le n \le 30{,}000$) and m ($0 \le m \le 100{,}000$), where n is the number of troop types and m is the total amount of money you have to spend.

Each of the next n lines will hold three values separated by spaces, representing a type of troop:

c h p

Where c ($1 \le c \le 100{,}000$) is an integer, which is the cost of that type of troop, h ($0.0 \le h \le 1.0$) is a real number, which is the health of that type of troop, and p ($0.0 \le p \le 1.0$) is a real number, which is the potency of that type of troop.

Output

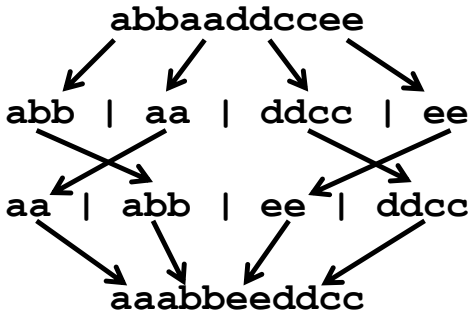
Output a single real number, which is the maximum strength you can achieve with your monetary resources. Output this number rounded to exactly 2 decimal places.

Sample Input	Sample Output
4 100000 300 1 0.02 500 0.2 1 250 0.3 0.1 1000 1 0.1	19436.05
2 100 1 0.1 1 1 1 0.1	3025.00

Paper Strips

Tito has been given a paper strip with a string of letters written on it. He would like to rearrange the letters. He does this by making some number of cuts between letters and then rearranging the strips of paper.

Tito likes order, so he would like the resulting strip of paper to be *bitonic*. That is, there should be some character position in the resulting string where the characters up to and including that position are alphabetically non-decreasing and all characters after and including that position are alphabetically non-increasing. Consider this example:



The resulting string in the above example is *bitonic*. Consider the first **e**. The string **aaabbe** is non-decreasing, and the string **eeddcc** is non-increasing. Tito achieved this with three cuts. Note that any string which is *monotonic* (uniformly nondecreasing or nonincreasing) is also *bitonic*.

Determine the minimum number of cuts that Tito needs in order to make his string *bitonic*.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will consist of a single line containing a string ***s*** ($1 \le |s| \le 50$) which consists only of lower-case letters. This is the original string on the strip of paper given to Tito.

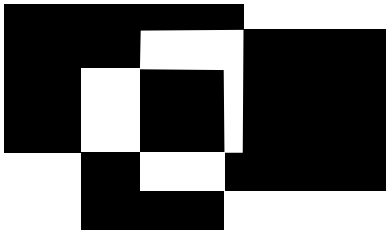
Output

Output a single integer, which is the minimum number of required cuts.

Sample Input	Sample Output
abbaaddccee	3

Rectangles

You are working on a new graphics system, which has added a new feature. Whenever you draw a figure, all the pixels in that figure flip from white to black, or from black to white. This image is what happens when three overlapping rectangles are drawn on a white field:



Starting with a white field, given a series of axis-aligned rectangles, how many pixels end up black?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with a single integer ***n*** ($1 \le n \le 100{,}000$) indicating the number of rectangles.

Each of the next ***n*** lines will have four space-separated integers ***x1***, ***y1***, ***x2*** and ***y2*** ($0 \le x1 < x2 \le 10^9$, $0 \le y1 < y2 \le 10^9$) which represent opposite corners of a rectangle. The rectangle consists of all pixels (**x**,**y**) such that $x1 \le x < x2$ and $y1 \le y < y2$, so the area of the rectangle is $(x2 - x1) \times (y2 - y1)$ pixels.

Output

Output a single integer, which is the number of pixels that are black after all of the rectangles are drawn on a white field.

Sample Input	Sample Output
2 0 0 4 4 1 1 3 3	12
4 0 0 10 10 1 1 11 11 2 2 12 12 3 3 13 13	72

Area Rug

The main room of your home is square, ***n*×*n*** feet. Unfortunately, the floor is dirty. You're a college student, so you hate to clean! Rather than clean it, you buy an area rug ***s*×*s*** feet square to cover some of the dirty spots.

Consider all of the ways that you could place the ***s*×*s*** area rug in the ***n*×*n*** room so that all ***s*×*s*** square feet of it cover part of the floor, axis aligned (no rotation). How many ways are there to cover a certain number of dirty spots?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with two space-separated integers ***n*** ($1 \le n \le 1{,}000$) and ***s*** ($1 \le s \le \min[n,100]$), where ***n*** is the size of one side of the room, and ***s*** is the size of one side of the new area rug.

Each of the next ***n*** lines will have a string of exactly ***n*** characters, consisting only of '**C**' (a clean spot on the floor) or '**D**' (a dirty spot on the floor).

Output

For each count of dirty floor spots covered, from 0 to s^2, if the number of ways of covering that many dirty spots with an area rug of size ***s*×*s*** is greater than 0, output the number of spots and the number of ways of covering them on a line, separated by a space. Output them in order, smallest number of dirty spots to largest.

Sample Input	Sample Output
10 5 DDDDDDDDDD DCCCCCCCCD DCCCCCCCCD DCCCCCCCCD DCCCCCCCCD DCCCCCCCCD DCCCCCCCCD DCCCCCCCCD DCCCCCCCCD DDDDDDDDDD	0 16 5 16 9 4

In this example, there are 4 ways to cover 9 dirty spots (the corners), 16 ways to cover 5 dirty spots (the non-corner edges), and 16 ways to cover 0 dirty spots (the interior).

To Tell the Truth

There are ***n*** people in a room, each of whom always tells the truth, or always lies. Each of them makes a statement of the form: "Some number between ***a*** and ***b*** (inclusive) of us are telling the truth." What is the maximum number of truth-tellers in the room?

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line containing an integer ***n*** ($1 \le n \le 1{,}000$) which is the number of people in the room.

Each of the next ***n*** lines will have two space-separated integers, ***a*** and ***b*** ($0 \le a \le b \le n$). Each line represents the statement of one person that "Some number between ***a*** and ***b*** (inclusive) of us are telling the truth."

Output

Output a single integer, which is the largest possible number of truth-tellers, or **-1** if the statements are inconsistent.

Sample Input	Sample Output
3 1 1 2 3 2 2	2
8 0 1 1 7 4 8 3 7 1 2 4 5 3 7 1 8	-1