

- A - [Flip Game](#)
- B - [Buffer Manager](#)
- C - [Triathlon](#)
- D - [Domino Puzzle](#)
- E - [Binary Search](#)
- F - [Frontier](#)
- G - [Garland](#)
- H - [Disk Tree](#)
- Z - [Sum](#)

2000-2001 ACM Northeastern European Regional Programming Contest

Problem A

"Flip Game"

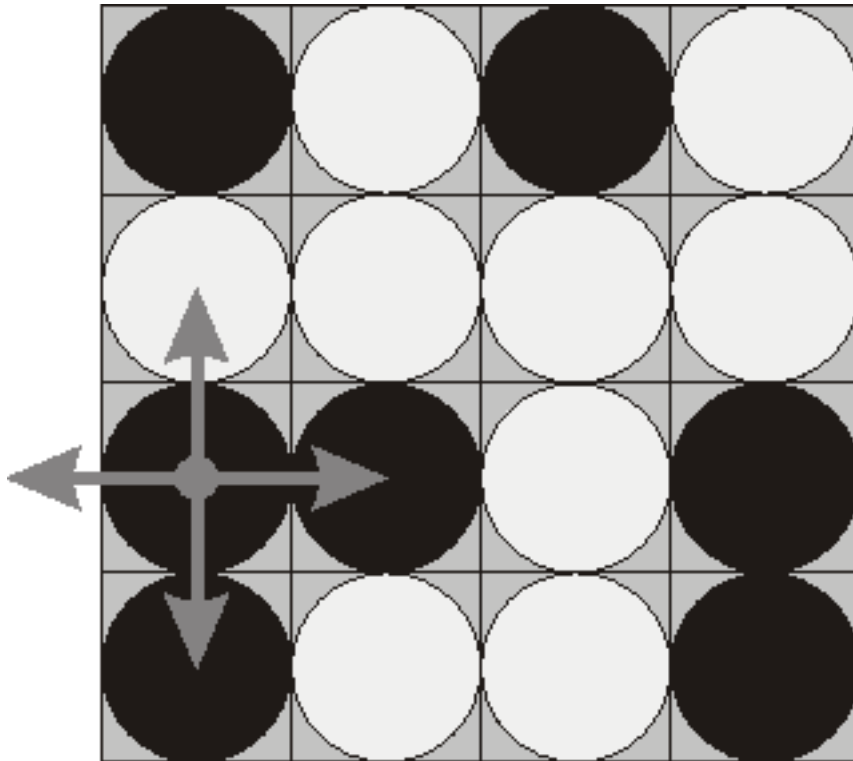
Input file FLIP.IN

Output file FLIP.OUT

Time limit 15 seconds per test

Flip game is played on a rectangular 4x4 field with two-sided pieces placed on each of its 16 squares. One side of each piece is white and the other one is black and each piece is lying either it's black or white side up. Each round you flip 3 to 5 pieces, thus changing the color of their upper side from black to white and vice versa. The pieces to be flipped are chosen every round according to the following rules:

1. Choose any one of the 16 pieces.
2. Flip the chosen piece and also all adjacent pieces to the left, to the right, to the top, and to the bottom of the chosen piece (if there are any).



Consider the following

position as an example:

```

bwbw
www
bbwb
bwwb

```

Here "b" denotes pieces lying their black side up and "w" denotes pieces lying their white side up. If we choose to flip the 1st piece from the 3rd row (this choice is shown at the picture), then the field will become:

```

bwbw
bwww
wwwb
wwwb

```

The goal of the game is to flip either all pieces white side up or all pieces black side up. You are to write a program that will search for the minimum number of rounds needed to achieve this goal.

Input

The input file consists of 4 lines with 4 characters "w" or "b" each that denote game field position.

Output

Write to the output file a single integer number - the minimum number of rounds needed to achieve the goal of the game from the given position. If the goal is initially achieved, then write 0. If it's impossible to achieve the goal, then write the word "Impossible" (without quotes).

Sample input #1

```
bwbw  
www  
bbwb  
bwwb
```

Output for the sample input #1

```
Impossible
```

Sample input #2

```
bwwb  
bbwb  
bwwb  
bwww
```

Output for the sample input #2

```
4
```

2000-2001 ACM Northeastern European Regional Programming Contest

Problem B

"Buffer Manager"

Input file BUFFER.IN

Output file BUFFER.OUT

Time limit 15 seconds per test

DBMS (Data Base Management System) development team has been successful in designing efficient Lock Manager and is going to proceed further. As a part of the team you will be responsible for the **Buffer Manager**.

Data blocks being read by DBMS from the hard drive are stored in the main memory in a fixed number of pre-allocated **buffers**. Each buffer can hold one data block. Each buffer can be either **free** (does not contain any useful information) or **occupied** by some data. When DBMS is going to read data block from the hard drive it has to decide which buffer to use for data storing. If there are any free buffers, then one of them is used for that purpose. If there are no free buffers, then one of the occupied buffers has to be flushed to become free, unless it was **locked** by some part of DBMS.

The choice of the buffer to flush is critical to DBMS performance. A lot of different algorithms were developed, LRU (Least Recently Used) algorithm being the one used most often. However, your DBMS is going to implement the Advanced Buffer Management algorithm which takes advantage of the fact that maximal performance is achieved when a number of consecutive data blocks from the hard drive are read into consecutive memory buffers.

Buffers are numbered from 1 to N , where N ($1 \leq N \leq 100000$) is a total number of buffers. Each buffer can be in any one of the following states: free, occupied or locked. Each occupied buffer is assigned an integer number from 1 to 9 – the **worthiness** of the currently stored information in that buffer. The worthiness of free buffers is considered to be zero. Locked buffers cannot be neither used nor flushed and their worthiness is undefined.

Having received the request to read K ($1 \leq K \leq 10000$) data blocks from the hard drive, Buffer Manager has to choose K consecutive non-locked buffers numbered from L to $L+K-1$ that have minimal possible sum of their worthiness, or to report that it is impossible to find K consecutive non-locked buffers. The latter can also happen if total number of buffers is less than K .

Your task is to write a program that models the processing of one request to Buffer Manager using the above algorithm.

Input

The first line of the input file contains two integers, N and K , separated by a space.

Starting from the second line there is a description of a buffers' state. The state of each buffer is represented by a single character:

- 0 – when the corresponding buffer is free.
- 1 – when the corresponding buffer is occupied and has worthiness of 1.
- 2 – when the corresponding buffer is occupied and has worthiness of 2.
- ...
- 9 – when the corresponding buffer is occupied and has worthiness of 9.
- * – when the corresponding buffer is locked.

Those characters are situated on the consecutive lines grouped by 80 characters per line without any spaces. Thus, each line starting from the second one contains exactly 80 characters with a possible exception for the last line.

Output

Write to the output file the single integer number L . This number gives the buffer number where first of the K blocks from the hard drive shall be read to ensure the minimal possible total worthiness of the blocks that have to be flushed. If there are more than one such value for L , then write the smallest one.

Write to the output file a single number 0 if it's impossible to find K consecutive non-locked buffers.

Sample input #1

```
100 53
2165745216091853477755800393859785807207523169954341**7363*9*94664808*4777717089
09825185827659480548
```

Output for the sample input #1

```
0
```

Sample input #2

```
100 10
2165745216091853477755800393859785807207523169954341**7363*9*94664808*4777717089
09825185827659480548
```

Output for the sample input #2

```
36
```

2000-2001 ACM Northeastern European Regional Programming Contest

Problem C

"Triathlon"

Input file TRIATH.IN

Output file TRIATH.OUT

Time limit 15 seconds per test

Triathlon is an athletic contest consisting of three consecutive sections that should be completed as fast as possible as a whole. The first section is swimming, the second section is riding bicycle and the third one is running.

The speed of each contestant in all three sections is known. The judge can choose the length of each section arbitrarily provided that no section has zero length. As a result sometimes she could choose their lengths in such a way that some particular contestant would win the competition.

Input

The first line of the input file contains integer number N ($1 \leq N \leq 100$), denoting the number of contestants. Then N lines follow, each line contains three integers V_i , U_i and W_i ($1 \leq V_i, U_i, W_i \leq 10000$), separated by spaces, denoting the speed of i^{th} contestant in each section.

Output

For every contestant write to the output file one line, that contains word "Yes" if the judge could choose the lengths of the sections in such a way that this particular contestant would win (i.e. she is the only one who would come first), or word "No" if this is impossible.

Sample input

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

Output for the sample input

```
Yes
Yes
Yes
No
No
No
Yes
No
Yes
```

2000-2001 ACM Northeastern European Regional Programming Contest

Problem D

"Domino Puzzle"

Dominoes

game played with small, rectangular blocks of wood or other material, each identified by a number of dots, or pips, on its face. The blocks usually are called bones, dominoes, or pieces and sometimes men, stones, or even cards.

The face of each piece is divided, by a line or ridge, into two squares, each of which is marked as would be a pair of dice...

The principle in nearly all modern dominoes games is to match one end of a piece to another that is identically or reciprocally numbered.

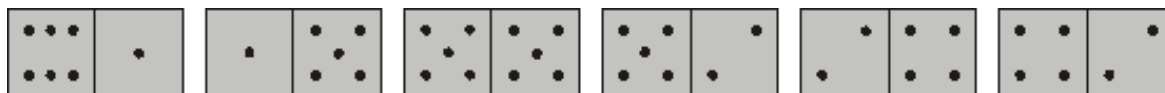
ENCYCLOPÆDIA BRITANNICA

Input file DOMINO.IN

Output file DOMINO.OUT

Time limit 15 seconds per test

Consider an arbitrary set of domino pieces where each piece is marked with two digits from 1 to 6. Some sets can be completely laid out in a row matching one end of a piece to another that is identically numbered, while others cannot. For example, the set consisting of 5 pieces: (1, 5), (1, 6), (5, 5) and (2, 4) twice, cannot be laid out in a row. However, if we add (2, 5) piece to the above set we could lay out the resulting set in the following row:



However, we are interested in a row having the smallest sum of digits on its pieces. In our example, instead of the piece (2, 5) with a sum of 7, we could add two pieces (1, 2) with a total sum of 6 to lay out the following row:



Your task is to write a program that for a given domino set will find an additional (possibly empty) set with the smallest possible sum of digits, so that a row could be laid out with both sets combined.

Input

The first line of the input file contains a single integer N ($2 \leq N \leq 100$) representing the total number of pieces in the domino set. The following N lines describe pieces. Each piece is represented on a separate line in a form of two digits from 1 to 6 separated by a space. The digits of a piece can be written in any order.

Output

On the first line of the output file write the smallest sum of digits of the additional set or 0 if that set is empty. On the second line write the total number of pieces in the additional set or 0 if that set is empty. Then write the pieces of the additional set in the same format as in input.

If there are a number of additional sets with the same smallest sum of digits exist then write any one of them.

Sample input #1

6
6 1
1 5
5 5
5 2
2 4
4 2

Sample input #2

5
1 5
6 1
5 5
2 4
2 4

Output for the sample input #1

0
0

Output for the sample input #2

6
2
1 2
1 2

2000-2001 ACM Northeastern European Regional Programming Contest

Problem E

"Binary Search"

Input file SEARCH.IN

Output file SEARCH.OUT

Time limit 15 seconds per test

The program fragment below performs binary search of an integer number in an array that is sorted in a nondescending order:

Pascal (file "sproc.pas")

```

const
  MAXN = 10000;
var
  A: array[0..MAXN-1] of integer;
  N: integer;

procedure BinarySearch(x: integer);
var
  p, q, i, L: integer;
begin
  p := 0;    { Left border of the search }
  q := N-1; { Right border of the search }
  L := 0;    { Comparison counter }
  while p <= q do begin
    i := (p + q) div 2;
    inc(L);
    if A[i] = x then begin
      writeln('Found item i = ', i,
        ' in L = ', L, ' comparisons');
      exit
    end;
    if x < A[i] then
      q := i - 1
    else
      p := i + 1
  end
end;

```

C (file "sproc.c")

```

#define MAXN 10000

int A[MAXN];
int N;

void BinarySearch(int x)
{
  int p, q, i, L;

  p = 0; /* Left border of the search */
  q = N-1; /* Right border of the search */
  L = 0; /* Comparison counter */
  while (p <= q) {
    i = (p + q) / 2;
    ++L;
    if (A[i] == x) {
      printf("Found item i = %d"
        " in L = %d comparisons\n", i, L);
      return;
    }
    if (x < A[i])
      q = i - 1;
    else
      p = i + 1;
  }
}

```

Before BinarySearch was called, N was set to some integer number from 1 to 10000 inclusive and array A was filled with a nondescending integer sequence.

It is known that the procedure has terminated with the message "Found item i = XXX in L = XXX comparisons" with some known values of i and L.

Your task is to write a program that finds all possible values of N that could lead to such message. However, the number of possible values of N can be quite big. Thus, you are asked to group all consecutive Ns into intervals and write down only first and last value in each interval.

Input

The input file consists of a single line with two integers i and L ($0 \leq i < 10000$ and $1 \leq L \leq 14$), separated by a space.

Output

On the first line of the output file write the single integer number K representing the total number of intervals for possible values of N . Then K lines shall follow listing those intervals in an ascending order. Each line shall contain two integers A_i and B_i ($A_i \leq B_i$) separated by a space, representing first and last value of the interval.

If there are no possible values of N exist, then the output file shall contain the single 0.

Sample input #2

Sample input #1

10 3

9000 2

Output for the sample input #2

Output for the sample input #1

4
12 12
17 18
29 30
87 94

0

2000-2001 ACM Northeastern European Regional Programming Contest

Problem F

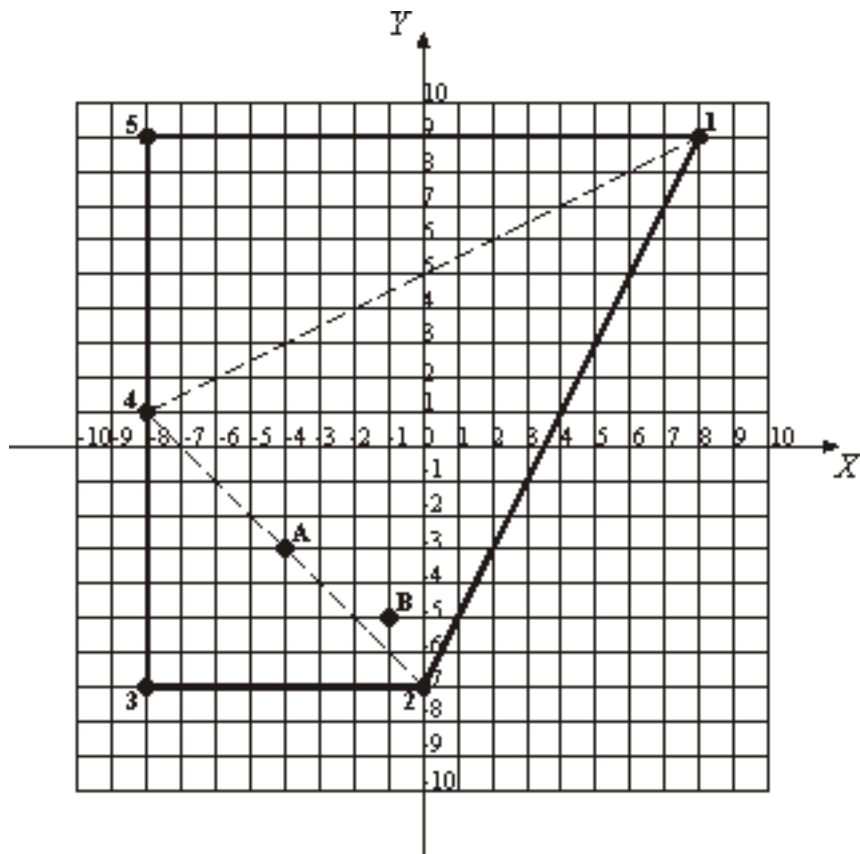
"Frontier"

Input file FRONTIER.IN

Output file FRONTIER.OUT

Time limit 15 seconds per test

Lilliputian frontier is a convex polygon with non-zero area. The vertices of this polygon are guard towers, which are connected by straight lines. This frontier is too long and expensive for Lilliputia to maintain; therefore the Lilliputian government has decided to revise it to make it shorter. However, they don't want to build new guard towers, but to use existing ones as a part of a new frontier.



Each day frontier guards inspect the frontier. They go from one guard tower to the next one, traversing the frontier clockwise. Guard towers are numbered from 1 to N according to this inspection order. Frontier revision should not change this way of inspection and the area of Lilliputia shall remain non-zero.

For example, the frontier that is shown on the picture (axes are in kilometer scale) is traversed by 1 - 2 - 3 - 4 - 5 - 1 route, which is 57.89 kilometers long. To make the frontier as short as possible Lilliputia should revise it so that the frontier is traversed by 2 - 3 - 4 - 2 route, thus reducing its length to 27.31 kilometers.

However, Lilliputia has a number of historical monuments which are its major pride. The historical monuments shall be kept inside Lilliputia at any cost and they should not end up on the frontier. So, the task is to design the shortest frontier that will preserve all historical monuments inside Lilliputia.

On the sample picture two historical monuments marked "A" and "B" are shown. The desire to keep them inside Lilliputia will lead to the shortest frontier with a traverse path 1 - 2 - 3 - 4 - 1 having 51.78 kilometers in length.

Input

The first line of the input file contains two integers N and M , separated by a space. N ($3 \leq N \leq 50$) is a total number of guard towers on the Lilliputian frontier. M

($0 \leq M \leq 1000$) is a total number of historical monuments that are situated inside Lilliputia.

Next N lines contain guard towers' coordinates in a clockwise order followed by M lines that contain historical monuments' coordinates. All coordinates are represented as two integers (for X and Y correspondingly) separated by a space. Coordinates are given in a kilometer scale and each coordinate does not exceed 10000 by an absolute value. All guard towers are located at distinct points.

Output

Write to the output file a single real number – the minimal possible length of the Lilliputian frontier (in kilometers) accurate to two digits to the right of the decimal point.

Sample input #1

```
5 0
8 9
0 -7
-8 -7
-8 1
-8 9
```

Output for the sample input #1

```
27.31
```

Sample input #2

```
5 2
8 9
0 -7
-8 -7
-8 1
-8 9
-4 -3
-1 -5
```

Output for the sample input #2

```
51.78
```

2000-2001 ACM Northeastern European Regional Programming Contest

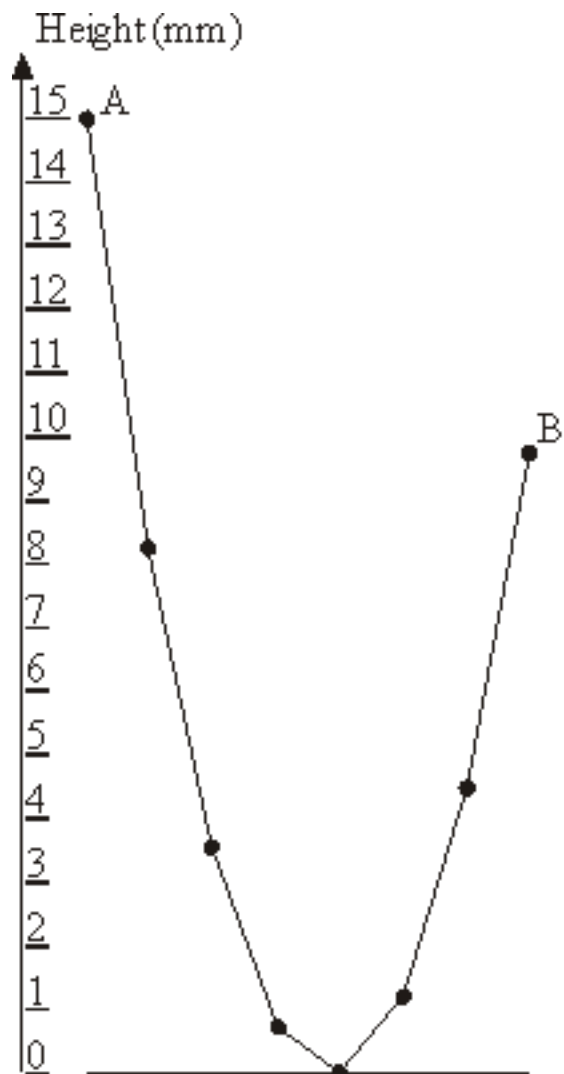
Problem G

"Garland"

Input file GARLAND.IN

Output file GARLAND.OUT

Time limit 15 seconds per test



The New Year garland consists of N lamps

attached to a common wire that hangs down on the ends to which outermost lamps are affixed. The wire sags under the weight of lamp in a particular way: each lamp is hanging at the height that is 1 millimeter lower than the average height of the two adjacent lamps.

The leftmost lamp is hanging at the height of A millimeters above the ground. You have to determine the lowest height B of the rightmost lamp so that no lamp in the garland lies on the ground though some of them may touch the ground.

You shall neglect the lamp's size in this problem. By numbering the lamps with integers from 1 to N and denoting the i^{th} lamp height in millimeters as H_i we derive the following equations:

- $H_1 = A$
- $H_i = (H_{i-1} + H_{i+1})/2 - 1$, for all $1 < i < N$
- $H_N = B$
- $H_i \geq 0$, for all $1 \leq i \leq N$

The sample garland with 8 lamps that is shown on the picture has $A = 15$ and $B = 9.75$.

Input

The input file consists of a single line with two numbers N and A separated by a space. N ($3 \leq N \leq 1000$) is an integer representing the number of lamps in the garland, A ($10 \leq A \leq 1000$) is a real number representing the height of the leftmost lamp above the ground in millimeters.

Output

Write to the output file the single real number B accurate to two digits to the right of the decimal point representing the lowest possible height of the rightmost lamp.

Sample input #1

8 15

Output for the sample input #1

9.75

Sample input #2

692 532.81

Output for the sample input #2

446113.34

2000-2001 ACM Northeastern European Regional Programming Contest

Problem H

"Disk Tree"

Input file DISKTREE.IN

Output file DISKTREE.OUT

Time limit 15 seconds per test

Hacker Bill has accidentally lost all the information from his workstation's hard drive and he has no backup copies of its contents. He does not regret for the loss of the files themselves, but for the very nice and convenient directory structure that he had created and cherished during years of work. Fortunately, Bill has several copies of directory listings from his hard drive. Using those listings he was able to recover full paths (like "WINNT\SYSTEM32\CERTSRV\CERTCO~1\X86") for some directories. He put all of them in a file by writing each path he has found on a separate line. Your task is to write a program that will help Bill to restore his state of the art directory structure by providing nicely formatted directory tree.

Input

The first line of the input file contains single integer number N ($1 \leq N \leq 500$) that denotes a total number of distinct directory paths. Then N lines with directory paths follow. Each directory path occupies a single line and does not contain any spaces, including leading or trailing ones. No path exceeds 80 characters. Each path is listed once and consists of a number of directory names separated by a back slash ("\").

Each directory name consists of 1 to 8 uppercase letters, numbers, or the special characters from the following list: exclamation mark, number sign, dollar sign, percent sign, ampersand, apostrophe, opening and closing parenthesis, hyphen sign, commercial at, circumflex accent, underscore, grave accent, opening and closing curly bracket, and tilde ("!#\$%&'()-@^_`{|}~").

Output

Write to the output file the formatted directory tree. Each directory name shall be listed on its own line preceded by a number of spaces that indicate its depth in the directory hierarchy. The subdirectories shall be listed in lexicographic order immediately after their parent directories preceded by one more space than their parent directory. Top level directories shall have no spaces printed before their names and shall be listed in lexicographic order. See sample below for clarification of the output format.

Sample input

```
7
WINNT\SYSTEM32\CONFIG
GAMES
WINNT\DRIVERS
HOME
WIN\SOFT
GAMES\DRIVERS
WINNT\SYSTEM32\CERTSRV\CERTCO~1\X86
```

Output for the sample input

```
GAMES
  DRIVERS
HOME
WIN
  SOFT
WINNT
  DRIVERS
  SYSTEM32
    CERTSRV
      CERTCO~1
        X86
  CONFIG
```

2000-2001 ACM Northeastern European Regional Programming Contest

Problem Z

"Sum"

Input file SUM.IN

Output file SUM.OUT

Time limit 15 seconds per test

Your task is to find the sum of all integer numbers lying between 1 and N inclusive.

Input

The input file consists of a single integer N that is not greater than 10000 by its absolute value.

Output

Write to the output file a single integer number that is the sum of all integer numbers lying between 1 and N inclusive.

Sample input

-3

Output for the sample input

-5