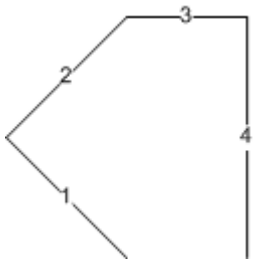
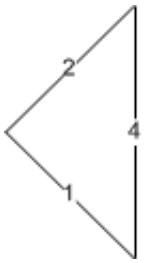
Mario got a new portable navigational device. This one special because it not meant to be used for find the shortest route. It is used for a Sunday walk through the nearby beautiful park.

Every day the device proposes a route for a walk. A route is composed of several segments. Each segment is described with two integers (X, Y), the relative movement position.

For example, consider the proposed route in the left illustration that is composed of four segments. First segment is described as (-1, 1), second segment as (1, 1), third as (1, 0) and finally fourth segment as (0, -2).

The final position of the proposed route finishes the walk 1 meter east of the starting position.





Mario doesn't want to finish the walk too far from the starting position, so he decided to remove exactly one segment. He will pick a segment so that the new route finishes as close as possible to the starting position. For example, if Mario removes third segment of the proposed route (as in the right illustration) he will finish exactly on the starting position.

Write the program that will calculate the final position of the proposed route, and will determine the shortest possible distance from start to finish if he removes one segment of the route.

INPUT

First line contains one N ($1 < N \leq 30$), the number of segments on the proposed route.

The next N lines contain two integers each X and Y ($-1000 \leq X,Y \leq 1000$) describing a route segment.

OUTPUT

In the first line output two integers X and Y that describe the final position of the proposed route.

In the second line output one real number (rounded to two decimal places), the shortest possible distance from start to finish if Mario removes one segment of the route.

SAMPLE TEST CASES

input

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

output

```
1 0
0.00
```

input

```
3
0 1
1 0
-1 -1
```

output

```
0 0
1.00
```

input

```
4
0 5
5 0
0 -5
-5 0
```

output

```
0 0
5.00
```

Clarification for the first sample: This sample corresponds to illustrations on the previous page (third segment is removed).

Clarification for the second sample: Removing first or second segment results in distance of 1.00. Removing the third segment results in distance of 1.41.

Clarification for the third sample: The distance is always equal to 5.00, no matter what segment he removes.

Social networks on the Internet became a part of the everyday life. One of the interesting phenomena is that the number of friendship is increasing rapidly. According to the old "a friend of a friend is my friend", each person every day checks the list of friends of his friends and adds new people to his list. It takes 1 day to confirm the new friendship. So, if A and B and friends then person A sees only friendships of person B that were made yesterday or before.

All friendships are symmetric – if A is a friend of B then B is a friend of A.

As social network we are observing in this task doesn't allow friendships to be broken at some point everyone will be friend to everyone.

Write a program that will determine the number of days until that happens. For each day output the number of new friendships that were made that day.

INPUT

The first line contains two integers N and M ($1 \leq N \leq 50$, $1 \leq M \leq N*(N-1)/2$), the number of users and initial number of friendships.

The next M lines contain two numbers each A and B ($1 \leq A \leq N$, $1 \leq B \leq N$, $A < B$) that describe friendships between users A and B. Test data will be such that solution will always exist.

OUTPUT

In the first line output one integer, the number of days K until everyone will be friend to everyone.

In each of the next K lines output one integer, the number of friendships that were made that day.

SAMPLE TEST CASES

input

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

output

```
1
1
```

input

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

output

```
2
3
3
```

input

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

output

```
1
6
```

Perica got an antique dot matrix printer that he can be control using his personal computer. While playing with his new old printer he discovered that printer recognizes three types of commands:

- `SET(X)` – This command sets character X into main memory.
- `NEXT(X)` – This command sets character X into secondary memory.
- `WRITE` – This command writes a character on the paper. The character to bi printed is read from the main memory unless previous command was `NEXT`. In that case, character is read from the secondary memory.

For example, next 8 commands will write "AABAA" on the paper:

```
SET(A), WRITE, WRITE, SET(B), WRITE, SET(A), WRITE, WRITE.
```

After a lot of thinking, Perica discovered that he could also do it in 7 commands:

```
SET(A), WRITE, WRITE, NEXT(B), WRITE, WRITE, WRITE.
```

Given a sequence of characters, write a program that will calculate the minimum number of commands needed to print that sequence.

Note: First command has to be `SET`.

INPUT

The first line contains the sequence of characters to be printed. The sequence contains upper case english letters and is not longer than 10 000.

OUTPUT

Output one integer, the minimum number of commands needed to print the sequence.

SCORING

Test data worth 25 points has the length od the sequence less than 25.

SAMPLE TEST CASES

input	**input**	**input**
NN	IOIX	BABCBACA
output	**output**	**output**
3	7	13

Clarification for the first sample: `SET(N), WRITE, WRITE.`

Clarification for the second sample: `SET(I), WRITE, NEXT(O), WRITE, WRITE, SET(X), WRITE.`

Clarification for the third sample: `SET(B), WRITE, NEXT(A), WRITE, WRITE, NEXT(C), WRITE, WRITE, SET(A), WRITE, NEXT(C), WRITE, WRITE.`