Problem	SONG	JANICA	LETTERS	TRAM
Executable file	song.exe	janica.exe	letters.exe	tram.exe
Source file	song.pas song.c song.cpp	janica.pas janica.c janica.cpp	letters.pas letters.c letters.cpp	tram.pas tram.c tram.cpp
Input file	song.in	janica.in	letters.in	tram.in
Output file	song.out	janica.out	letters.out	tram.out
Time limit per test	10 seconds	10 seconds	10 seconds	10 seconds
Number of tests	10	10	10	10
Points per test	3	4	6	7
Total points	**30**	**40**	**60**	**70**
	200			

SONG

A song consists of one or more verses, and each verse consists of **four** lines. Each line consists of one or more words separated by single blank character, and finally, each word consists of one or more letters (a-z, A-Z).

We define the **last syllable** of a word to be the sequence of letters from the **last vowel** (inclusively) to the end of the word. If a word has no vowel, then the last syllable is the word itself.

We say that two lines **rhyme** if they have same last syllables (ignoring the letter case). Verse can have **perfect rhyme, even rhyme, cross rhyme, shell rhyme** or there can be no rhyme at all (**free rhyme**).

Verse has a **perfect rhyme** if all lines in a verse mutually rhyme (**a a a a**).

If verse does not have a perfect rhyme then we say that it has:

- **even rhyme:** if the first and second line rhyme and also third and fourth line rhyme (**a a b b**).
- **cross rhyme:** if the first and third line rhyme and also second and fourth line rhyme (**a b a b**).
- **shell rhyme:** if the first and fourth line rhyme and also second and third line rhyme (**a b b a**).

Write a program that will determine the rhyme for each verse in a song.

Input data

The first line of the input file contains an integer N, the number of verses in the song, $1 \leq N \leq 5$.

The following 4N lines of the input file contain the lines of the song. Maximal length of each line is 50.

Output data

Output file should have N lines. For each verse in a song there should a single line containing one of the words '**perfect**', '**even**', '**cross**', '**shell**' or '**free**' describing the rhyme in that verse.

Examples

song.in

```
1
mi smo Super
Edit Pjaf Ran
gari geri kuper
ej pazi na ekran
```

song.out

```
cross
```

song.in

```
2
Tko je to
to je Zlo
nije Zlo
to je Mo
Gdje je Mu
jel na putu
Brijem da nije
vidlo bi se
```

song.out

```
perfect
even
```

song.in

```
2
mirko me
ozbiljno zanima e
dok ovo pisem
drhtim placem
ali znam da i ja
mogu racunati
na to ma znate sram me
sapce da i ja njega mamim
```

song.out

```
even
free
```

Watching alpine skiing on TV can be really boring sometimes. Fortunately, it can happen that your TV operator doesn't show all the information about the race and then you have to do some computation to determine the winner yourself.

Here are some details about a race: Each race consists of two rounds. In first round, N skiers numbered from 1 to N, race starting in that order.

When the first skier in the first round (numbered with 1) finishes the race we are given the **time** he/she needed to complete the course. For every following skier we are given **time difference** between his/her time and the time of the skier **currently leading** the race.

Only the best M skiers qualify for the second round and they start the second round ordered by time from the first round decreasingly (last skier qualified for the second round starts first and the leader after the first round start last).

In the second round we are given the **total time** (first and second round added) of the first skier racing, and then for every following skier we are given the time difference between his/her total time and the total time of the skier currently leading the race.

Write a program that will **determine** medal winners after the race has finished. You can assume that it can never happen that two skiers have same time (after the first round or after both rounds).

Time for any race will be between 10 and 300 seconds. All times will be real numbers with at most two decimal digits.

Input data

The first line of the input file contains two integers N and M, separated by a single blank character, $3 \leq M \leq N \leq 100$.

The following line contains the time for the first skier in the first round, and the following N-1 lines contain the described time differences.

The following line contains the time for the first skier in the second round, and the following M-1 lines contain the described time differences.

Output data

The first line of the output file should contain the number of the skier winning the gold medal, second line the number of skier winning the silver medal, and the third line the number of skier winning the bronze medal.

Examples

janica.in

```
3 3
25.13
+1.14
+2.18
45.08
+2.14
+3.11
```

janica.out

```
3
2
1
```

janica.in

```
4 3
29.18
+2.18
+0.05
+1.13
54.22
+1.23
+1.11
```

janica.out

```
4
1
3
```

janica.in

```
5 4
27.29
-1.02
+1.83
-0.43
+0.03
56.98
+1.83
-0.43
+0.03
```

janica.out

```
5
4
1
```

LETTERS

A single-player game is played on a rectangular board divided in R rows and C columns. There is a single uppercase letter (A-Z) written in every position in the board.

Before the begging of the game there is a figure in the upper-left corner of the board (first row, first column). In every move, a player can move the figure to the one of the adjacent positions (up, down, left or right). Only constraint is that a figure **cannot visit** a position marked with the same letter twice. The goal of the game is to play **as many** moves **as possible**.

Write a program that will calculate the **maximal** number of positions in the board the figure can visit in a single game.

Input data

The first line of the input file contains two integers R and C, separated by a single blank character, $1 \leq R, S \leq 20$.

The following R lines contain S characters each. Each line represents one row in the board.

Output data

The first and only line of the output file should contain the maximal number of position in the board the figure can visit.

Examples

letters.in

```
2 4
CAAB
ADCB
```

letters.out

```
3
```

letters.in

```
3 6
HFDFFB
AJHGDH
DGAGEH
```

letters.out

```
6
```

letters.in

```
5 5
IEFCJ
FHFKC
FFALF
HFGCF
HMCHH
```

letters.out

```
10
```

TRAM

Tram network in Zagreb consists of a number of intersections and rails connecting some of them. In every intersection there is a switch pointing to the one of the rails going out of the intersection. When the tram enters the intersection it can leave **only** in the direction the switch is pointing. If the driver wants to go some other way, he/she has to manually **change** the switch.

When a driver has do drive from intersection A to the intersection B he/she tries to choose the route that will minimize the number of times he/she will have to change the switches manually.

Write a program that will calculate the **minimal** number of switch changes necessary to travel from intersection A to intersection B.

Input data

The first line of the input file contains integers N, A and B, separated by a single blank character, $2 \leq N \leq 100$, $1 \leq A, B \leq N$, N is the number of intersections in the network, and intersections are numbered from 1 to N.

Each of the following N lines contain a sequence of integers separated by a single blank character. First number in the i-th line, K_i ($0 \leq K_i \leq N-1$), represents the number of rails going out of the i-th intersection. Next K_i numbers represents the intersections directly connected to the i-th intersection. Switch in the i-th intersection is initially pointing in the **direction** of the first intersection listed.

Output data

The first and only line of the output file should contain the target minimal number. If there is no route from A to B the line should contain the integer '-1'.

Examples

tram.in

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

tram.out

```
0
```

tram.in

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

tram.out

```
1
```

tram.in

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

tram.out

```
-1
```