24th BULGARIAN NATIONAL OLYMPIAD IN INFORMATICS

2008

This booklet and CD were prepared under the financial support of the Telerik Inc.

Telerik Inc. was the official sponsor of the Bulgarian National Olympiad in Informatics for the last two years.

Telerik Inc. is a leading vendor of development tools and components for Microsoft .NET. Developers from most of the Fortune 2000 companies, many of the world's leading educational and non-profit organizations and thousands of consultants rely on Telerik products to build applications with unparalleled richness, responsiveness and interactivity.

Telerik Inc. has won a number of prestigious awards honoring its entrepreneurial approach and dedication to innovations. At the end of 2007, Telerik was acknowledged with 3rd place in Deloitte's "Rising Star" Technology Fast 50 program for the 50 fastest growing technology companies in Central Europe in 2007. Telerik was also ranked #1 in the "Best Employers 2007 (Bulgaria)" study, conducted by Hewitt Associates - small to medium size company category.

In April, 2008, Telerik Inc. won the Red Herring 100 Europe for 2008. The award recognizes the top 100 private technology companies headquartered in Europe, the Middle East and Africa region each year, based on financial performance, quality of management, execution of strategy, and dedication to research and development.

Contents

Introduction

In 1985, a team of the Union of Bulgarian Mathematicians organized the first National Olympiad in Informatics (NOI) in Bulgaria. In the present days, NOI is organized by the Ministry of Education and Science with the scientific support of the Union of Bulgarian Mathematicians.

Style of the Competition

- Algorithmic style programming tasks (like the International Olympiad in Informatics). Tasks typically used in NOI are focused on designing and programming efficient and correct algorithms.
- Individual competition, presence needed (attendance of the participants at an appointed time and place), free of charge for contestants.
- The competition has a time limit for solving the tasks (3 to 5 hours).
- Method of evaluation: with postponed evaluation of the submitted source codes.
- Competition rounds: Round 1 (School Round) – open for everyone; Round 2 (Regional Round) – only by invitation, depending on the results of the Round 1; Round 3 (Final Round) – only by invitation, depending on the results of the Round 2;
- Age of participants: from 11 to 19 years
- Competition groups: A –12^{th} and 11^{th} grade, B – 10^{th} and 9^{th} grade, C – 7^{th} and 8^{th} grade, D – 6^{th} grade and E – 4^{th} and 5^{th} grade.
- All the competition materials: problems, results, authors' and contestants' solutions are published on the Internet.
- NOI is organized by the Ministry of Education and Science with the scientific support of the Union of Bulgarian Mathematicians. The National Committee in Informatics (NC) is responsible for the Olympiad. The chair and the members of NC are proposed by the Union of Bulgarian Mathematicians and are approved by the Minister of Education and Science. The competition problems are prepared by NC.
- Way of announcement of the competition: NOI is included in the annual schedule of the Ministry of Education and Science and is announced by the structures of the Ministry.
- Types of awards: the best students from the national round are admitted to the extended national team. The actual national teams are determined after subsequent control competitions. They participate in the Balkan Olympiad in Informatics and in the International Olympiad in Informatics. The best students from group A in the Final round award the title "Winner of the National Olympiad" and are admitted without entrance exams at the top Bulgarian universities.
- Financial Basis of the competition**:** The competition is supported financially by the Ministry of Education, the Union of Bulgarian Mathematicians and sponsors. The official sponsor of the National Olympiad in Informatics for the last two years was Telerik.

Round 1

Round 1 / Task A1. FRACTIONS

Write program **`fract`** that finds the sum of all non-reducible fractions $\frac{a}{b}$, where a and b are integers, for which $k \le a < b \le n$.

Input
The program has to read data from the standard input. In a single line, the values of integers k and n $(1 \le k < n \le 100,\ n - k < 10)$ are given.

Output
The only line of the standard output should contain a non-reducible fraction (two integers, separated by a slash '/') , that is equal to the described sum.

EXAMPLE

Input
```
1 3
```

Output
```
3/2
```

Round 1 / Task A2. POLYGON

Let us consider a regular polygon with *n* vertices, inscribed in a unit circle ($2 < n < 1000$). The vertices are numbered clockwise from 1 through *n*. Starting at vertex 1, we are moving clockwise making a straight line segment ending at a vertex that is placed after *p* consecutive vertices ($0 < p < n - 1$). So, going from vertex 1, we arrive at vertex $p + 2$. We continue the movement in the same manner, step by step, and stop when we arrive again at vertex 1. Write program **`gone`** that inputs *n* and *p*, and outputs the total traveled length. The output should be a value with a decimal point and precision of 5 digits after the decimal point.

EXAMPLES

Input
```
4 1
```

Output
```
4.00000
```

Input
```
4 2
```

Output
```
5.65685
```

Round 1 / Task A3. BELONGING

The set of positive integers ***M*** is constructed by the following rules:

1) $1 \in \boldsymbol{M}$.
2) If $\boldsymbol{a} \in \boldsymbol{M}$, let's denote with α its binary representation. Then the number with binary representation $\overline{11\alpha}$ belongs to ***M***, too.
3) Let $\boldsymbol{a} \in \boldsymbol{M}$ and $\boldsymbol{b} \in \boldsymbol{M}$, α and β being their binary representations respectively. Then the number with binary representation $\overline{\alpha 0 \beta}$ belongs to ***M***, too.

For example, the first 10 members of ***M*** in increasing order are 1, 5, 7, 21, 23, 29, 31, 85, 87, 93.

Write program **`belong`**, which determines whether each of the three input positive integers ***N***, P, and Q belongs to ***M*** or not.

Input

One line with three positive integers ***N***, P, and Q, separated by a space is read from the standard input. None of the numbers has more that 18 decimal digits.

Output

Send to the standard output one line with three symbols, each 0 or 1: for each input number (in the same order) write 0, if it doesn't belong to ***M***, or 1 if it does.

EXAMPLE

Input

```
1270 95 2047
```

Output

```
011
```

Round 1 / Task B1. MATH

Miss Peterson has noticed that her students have still difficulties adding and subtracting integers. Because of that, at the end of the last few periods, she has been writing n integers between -100 and 100 on the board. Students' homework assignment has been to insert the "+" and "–" signs and to calculate the expression. The students cannot change the positions of the integers and there must be k subtractions and $n-k-1$ additions. Miss Peterson, however, has promised to give an excellent grade only to those students who have not only calculated the expression correctly, but have also obtained the greatest possible number as an answer. Shortly before the end of the first term, Johnny desperately needs this excellent grade. He can add and subtract integers, but wants to be sure that he will obtain the greatest number. Help Johnny, writing the program **`math`** which finds this greatest value.

Input

The program has to read the input data from the standard input. In the first line, the values of n and k ($0 < k < n < 30$) are given. In the second line, the n numbers are given.

Output

The only line of the standard output should contain the found greatest value.

EXAMPLE

Input
```
3 1
-10 5 -6
```

Output
```
1
```

Round 1 / Task B2. SUM OF DIGITS

Let us consider a binary representation of positive integer n. We denote by $B(n)$ a sum of its digits. For example, for $n=19=10011_2$ we have $B(19)=3$. Write program **bsum** that determines the sum $S=B(1)+B(2)+\ldots+B(n)$ for a given n.

Input

A line with a single positive integer $n \leq 1000000$ is read by the standard input.

Output

The program has to write on the standard output one line containing the calculated sum S.

EXAMPLE

Input
```
387
```

Output
```
1612
```

Round 1 / Task B3. ARRANGING FOR A PARTY

It is a torment to arrange guests around the party table! Three adjacent women – and tittle-tattle is guaranteed! On the other hand, if a man has a woman on each side, it will be difficult to be gallant to both. You will receive M_1 men и N_1 women to be settled around the first table, and M_2 men и N_2 women to be settled around the second one. Write a program to help you solve the hard task. People listed for each table may not be mixed.

Input

Two line are read from the standard input, each containing two positive integers (number of men and number of women, in this order) separated by a space. The numbers do not exceed 100.

Output

Write to the standard output one line correspondingly for each of the tables: one solution of the problem or the message `NO`, if there is none. If there is a solution, describe it as a symbolic string, writing 0 for "woman" and 1 for "man", starting wherever you wish. Do not forget that the last guest you mention is in fact side by side with the first one – the tables are round!

EXAMPLE

Input
```
7 3
2 4
```

Output
```
1011100111
NO
```

Round 1 / Task C1. TRIANGLES

Write program **`triangles`** to calculate the number of different triangles with perimeter P such that, the lengths of the sides are integers, when P is an integer.

Input
The program has to read the input data from the standard input. In a single line, the value of P ($2 < P < 100000$) is given.

Output
The only line of the standard output should contain the calculated number of triangles.

EXAMPLE

Input
```
5
```

Output
```
1
```

Round 1 / Task C2. CLOCK

A digital clock shows hours and minutes from `00:00` through `23:59`. Write a program **`clock`** which, using two given readings of the clock display, calculates the time between these two readings.

Input
The program has to read data from the standard input. The first and the second reading of the clock display are given in two lines, respectively. The two readings are different, each of them containing five symbols – the first two denote the hour, the third is ":" and the last two denote the minutes.

Output
The only line of the standard output should contain the calculated time in the same form as the input data are given.

EXAMPLE

Input
```
23:58
01:02
```

Output
```
01:04
```

Round 1 / Task C3. MINIMAL DIFFERENCE

Four distinct non-zero decimal digits are given. Two two-digit numbers a and b are composed using each given digit exactly once. We suppose that $a > b$. Write program **`mindif`** to calculate the value of the minimal possible difference $a - b$.

Input
Your program should read from the standard input one line containing 4 distinct non-zero digits, separated by spaces.

Output
Your program should write to the standard output a line, containing the calculated minimal difference.

EXAMPLE

Input
```
5 2 9 1
```

Output
```
6
```

Explanation: The minimal difference in the example is obtained by subtracting 19 from 25.

Round 1 / Task D1. RECTANGLES

Write program **`rectangles`** to calculate the number of rectangles with area S cm^2, such that the lengths of the sides in cm are integers, when S is an integer.

Input
The program has to read the input data from the standard input. In a single line, the value of S ($1 < S < 100000$) is given.

Output
The only line of the standard output should contain the calculated number of rectangles.

EXAMPLE

Input
```
4
```

Output
```
2
```

Round 1 / Task D2. DIVISIBILITY BY 3

Three distinct non-zero digits are given. Write program **`least3`** that finds out the least positive integer divisible by 3 that has in its decimal representation digits among the given ones only. Each given digit may not be included or may be included several times in the result.

Input
Your program should read by the standard input one line containing three distinct non-zero decimal digits, separated by spaces.

Output
Your program should send to the standard output one line with the required number.

EXAMPLES

Input	**Input**
`4 7 2`	`7 1 4`
Output	**Output**
`24`	`111`

Round 1 / Task D3. APPROXIMATION

Write program **`prox`** that inputs three positive integers *a*, *b*, *c* (less than 999), and finds out an integer *d* which is the closest to c, and *d* can be obtained by applying addition or multiplication of *a* and/or *b*. If there is more than one such integer *d*, output the smallest one.

EXAMPLES

Input	**Input**	**Input**
`3 5 11`	`3 5 9`	`3 5 7`
Output	**Output**	**Output**
`10`	`9`	`6`

Explanation for Example 1: All possible results obtained by the operations, described in the task statement are 3+5, 3*5, 3+3, 5+5, 3*3, 5*5, i.e. 8, 15, 6, 10, 9, 25. Among them, the value 10 is the closest to 11.

Round 1 / Task E1. ANT

The last summer, Zdravka the Ant came across a big pile of *n* bread crumbs. The firs crumb was 1 cm away from the ant-hill, the second one – 2 cm away, the third one – 3 cm away, the fourth one – 4 cm away, …, the n^{th} – *n* cm away. Zdravka the Ant would come out the ant-

hill, take a crumb, bring it home in the ant-hill and then proceed in the same way until all crumbs were collected. Write a program **ant** which calculates the length of the course that Zdravka the Ant traveled to bring all crumbs to the ant-hill.

Input
The program has to read the input data from the standard input. In a single line the value of n ($1 < n < 250$) is given.

Output
The only line of the standard output should contain the length of the course in cm that Zdravka the Ant traveled.

EXAMPLE

Input
```
5
```

Output
```
30
```

Round 1 / Task E2. NUMBERS

Johnny knows only the first digit a and the last digit b of a three-digit number $\overline{a*b}$. Write program **numbers** which calculates the number of ways he can replace the asterisk with a digit, so that the obtained number is divisible by integer *k*.

Input
The program has to read data from the standard input. In a single line, the values of integers a, b and c ($0 < a < 10$, $0 \le b < 10$, $1 < k < 100$) are given.

Output
The only line of the standard output should contain the number of ways to replace the asterisk.

EXAMPLE

Input
```
1 2 2
```

Output
```
10
```

Round 1 / Task E3. RABBIT

Write program **rabbit** that outputs on the screen in a given order three figures described below. Each figure consists of 5 characters on a line and has assigned a key according to the following scheme:

Key 1 specifies the figure `(\_/)`

Key 2 specifies the figure (○.○)

Key 3 specifies the figure (_._)

Sometimes, your program may output a rabbit, but sometimes it may output another picture.

Input
Three integers on a row in the standard input, each integer equals to 1, 2, or 3. Some values of the input integers may be repeated.

Output
Your program has to output to the standard output 3 figures, each on the beginning of 3 separate consecutive lines. The kind of each figure has to be chosen according to the given key value in the input.

EXAMPLES

Input
```
1 2 3
```

Output
```
(\_/)
(○.○)
(_._)
```

Input
```
2 2 1
```

Output
```
(○.○)
(○.○)
(\_/)
```

Round 2

Round 2 / Task A1. SUMS OF PRIME NUMBERS

Let n be a positive integer. We denote by $S(n)$ the number of ways in which n can be written as a sum of two or more primes. For example $S(9) = 4$, because $9 = 2 + 2 + 2 + 3 = 2 + 2 + 5 = 2 + 7 = 3 + 3 + 3$. Let p be the smallest prime greater than n. Write a program **`primes`** which calculates the remainder of division of $S(n)$ by p.

Input
The program has to read data from the standard input. In a single line, the value of the positive integer n ($n < 50000$) is given.

Output
The only line of the standard output should contain the remainder of division of $S(n)$ by p.

EXAMPLE

Input
```
9
```

Output
```
4
```

Round 2 / Task A2. GROUPS

On a straight line, n distinct points are given with their coordinates $a_1 < a_2 < \ldots < a_n$. We split them into p consequently placed from left to right nonempty groups and compute the arithmetic mean s_i of point's coordinates for the i-th group, $i = 1, 2, \ldots, p$. Then for each point in i-th group we compute the difference in point's coordinate and the value of s_i. Let us denote by d_i the sum of squares of all such differences. Write program **`groups`**, that finds the minimal value of $d_1 + d_2 + \ldots + d_p$.

Input
On the first line in the standard input, the values of n and d ($0 < n < 500$, $0 < p < n$) are given. On the second line, the values of $a_1, a_2, \ldots, a_n$, are given in an increasing order as numbers with decimal point, containing at most 3 digit after the decimal point. All values are separated by spaces.

Output
On the standard output, your program has to output the value that we have to obtain as a number with a decimal point with precision of 10^{-4}.

EXAMPLE

Input
```
5 3
1.0 2.0 3.0 4.0 5.0
```

Output
```
1.0
```

Round 2 / Task A3. EXPRESSIONS

Given is an arithmetic expression composed of small letters, sings of operations: + (addition), – (subtraction), * (multiplication), / (division), ^ (raising to power), and brackets. To compute a value of the expression, we observe the following rules:

- Raising to power we perform before multiplication and division; in their turn, multiplication and division we perform before addition and subtraction.
- Several consecutive operations of subtraction or division we perform from left to right; the same we observe for several alternated operations of subtraction and addition, or division and multiplication. Several consecutive operations of raising to power we perform from right to left.
- We compute a part of the expression embraced in brackets before computing its neighbor operations.

Observing the above rules, we may compute a value of a given expression using several different orders to perform operations. For example, a sequence of several additions, we may compute in an arbitrary order. Another example: for every operation we may chose the order to compute its first and second operands.

Write program **`eval`** that finds how many ways are possible to compute correctly a given expression.

Your program has to read an arithmetic expression by a line from the standard input, and has to write an integer on the standard output which is equal to the number of all possible ways of computations.

The length of the input string is no greater then 25.

EXAMPLES

Input	**Input**
`a-b-c`	`(a+b)*(a-b)`
Output	**Output**
`1`	`2`

Round 2 / Task B1. DECIMAL FRACTION

Let *a* and *b* be natural numbers. Let us represent number $\frac{a}{b}$ as a decimal fraction. It can be terminating (e. g. $\frac{1}{4} = 0,25$) or non-terminating (e. g. $\frac{1}{3} = 0,333...$). We may assume any

terminating decimal fraction to be non-terminating by adding infinitely many 0's at the end (e. g. $\frac{1}{4} = 0,25000...$). Write program **dfrac** which, on given natural numbers *a*, *b*, *k* and *p*, obtains *p* consecuitive digits in the decimal representation of $\frac{a}{b}$, the first of them being the k^{th} digit after the decimal point.

Input
The program has to read data from the standard input. The values of *a* and *b* are given in the first line; the values of *k* and *p* are given in the second line ($0 < a < b < 30000000$, $0 < k < 10^{18}$, $0 < p < 50$).

Output
The only line of the standard output should contain the obtained digits.

EXAMPLE

Input
```
1 7
3 10
```

Output
```
2857142857
```

Round 2 / Task B2. PLATES

Given are *m* identical transparent square plates. On each plate is drawn the same rectangular net, containing *n* by *n* unit square cells. Some cells on some plates are marked. We assume two plates to be no distinct, if we can rotate or/and turn face down the first plate, so that it appears to have marked cells on the same places as the second plate. Write program **plates** that computes how many plates are contained in the largest subset of all given plates, in which any two plates are distinct.

Input
On the first line of the standard input, the values of *m* and *n* are given ($1 < m \leq 1000$, $1 < n < 30$). A description of each plate follows as a sequence of 0 and 1, line by line. A marked cell is denoted by 1, unmarked by 0. There are no spaces between 0's and 1's in a line. There are no empty lines between descriptions of the consecutive plates.

Output
On a single line of the standard output, your program has to write the computed number as an integer.

EXAMPLE

Input
```
3 2
01
10
```

```
10
01
11
10
```

Output
```
2
```

Round 2 / Task B3. STAMPS

We have to stick stamps of total value *S* onto a letter. In the post office, stamps are sold for *N* different values, the smallest one equal to 1 cent. There is an unlimited amount of stamps for every available value. Write program **`stamps`** to find the minimal number of stamps which are enough to obtain value *S*.

Input
On the first line of the standard input, the integer values of *S* and *N* are given ($0 < S < 5000$, $0 < N < 3000$). On the second line, *N* distinct values of stamps are given, i.e. values that are sold in the post office. These values are positive integers, each less or equal to *S*.

Output
On one line in the standard output, your program has to write an integer, equal to the found minimal number.

EXAMPLES

Input	**Input**
`22 5`	`1000 7`
`1 4 8 12 15`	`30 1 12 2 11 18 14`
Output	**Output**
`4`	`35`

Round 2 / Task C1. FRIENDS

Parvan belives himself a very experienced programmer because is participating for the third year in many programming contests. That is why he is ready to enter any contest and compete with opponent teams composed of students of his age. As a beginning, he composed a team of all students that he knows and that participate in the contests for his age group (called C in Bulgaria). All of his friends accepted the invitation, but decided that their own friends, which Parvan does not know, would like to enter the contest also and invited them to make a team – team of friends of friends of Parvan. The second team decided to make the same and has invited their friends that are not included in a team yet, to make a third team and so on, till all contestants of the world that had friendships relations were included in one of the teams. To be able to manage the process, they labeled each participant by a number from 1 through *N*, and established all friendships. Number 1, of course, was given to Parvan. The question is: how Parvan will compete with teams composed of 100 or 200 contestants, especially if some

of the teams contain 3-4 students of his high level. Write program **`friends`** to find the number of students in the largest team.

Input

On the first line of the standard input, separated by a single space, the number N of the contestants and the number M of the couples of friends are given ($N \le 2000$, $M \le 1\ 000\ 000$). Each of the next M lines contains, separated by a single space, the labels of two contestants that are friends.

Output

On the single line of the standard output, your program has to print the number of contestants in the largest team.

EXAMPLE

Input	**Output**
6 6 1 2 1 3 2 4 3 4 3 5 3 6	3

Round 2 / Task C2. ANGLES

The teacher of Parvan in mathematics likes to propose to students strange tasks. For example, she drew square table with N rows and N columns, indexed by the numbers from 1 through N, each cell of which contains one integer. For each cell C of the table, she defined its *angle*, as the set of K consecutive cells in the row of C, situated right of C, K consecutive cells in the column of C, situated below C and the cell C itself. If right of C and/or below it there is less than K cells, the angle of C contains all cells to the end of the row and/or all cells to the end of the column. The students had to find for each cell the sum of numbers in its angle and to decide which cell has an angle with the maximum sum. Write program **`angles`** to find the index of the row and the index of the column of the cell with the maximum sum of its angle.

Input

The first line of the standard input contains, separated by a single space, the numbers N and K ($N \le 1000$, $1 \le K \le N/2$). Each of the next N lines contains N integers, separated by single spaces – the contents of the cells from the corresponding row of the table. All these numbers are between –1000 and 1000.

Output

On the single line of the standard output, the program has to print separating by a single space the index of the row and the index of the column of the cell with the maximum sum of its angle. If there are several cells with the same maximum sum, the program has to print this one that has the minimal index of the row. If there are still several cells with the same maximum sum, the program has to print this one that has the minimum index of the column.

EXAMPLE

Input

```
6 3
1 2 6 9 -1 0
1 3 -5 0 2 8
6 2 2 -7 3 1
5 5 -3 7 -2 4
1 1 1 12 0 6
-2 3 7 10 2 1
```

Output

```
4 4
```

Round 2 / Task C3. JUSTIFY

Parvan is very talented in Informatics just as well in Information Technologies. Learning text processing in class, he formulated the following task: Each sequence of Latin or Cyrillic letters, digits or punctuation marks that does not contain spaces is called *word*. *Text* is composed of some number of *rows* – sequences of words separated by spaces. Alignment in given number of positions, for example 60, is usual operation of the text processing. A row of the text is *aligned from both sides* (*justified*) *in 60 positions*, when it starts and ends with a character different from a space, and missing characters to obtain a length of 60 are ***well distributed*** among the words of the row. Well distributed means that either all sequences of spaces are of the same length or, when this is impossible, some of the sequences of spaces are longer by one character than the others. Something more, all longer sequences of spaces are on the leftmost of all shorter sequences. Write program **`justify`**, which has to justify a given text in 60 positions.

Input

On the first line of the standard input, a positive integer *N* is given – the number of rows of the text that has to be justified. The length of each row is less than or equal to 60. The length of a word in the text is no greater than 29 characters. It could have more than one space between two words of the text.

Output

On the standard output, your program has to print all the given text, justified in 60 positions. Because the last row of the text could be very short, the words of the last row have to be separated by single spaces (i.e. the last row has to be left aligned).

EXAMPLE

Input

```
4
При   това, всички  по-дълги    последователности     са
  преди  всички  по-къси. Напишете    програма justify,
която   по      зададен текст го подравнява двустранно в 60
позиции на ред.
```

Output

```
При    това,    всички    по-дълги    последователности   са
преди    всички    по-къси.   Напишете   програма   justify,
която по зададен текст го подравнява двустранно в 60
позиции на ред.
```

Round 2 / Task D1. RING

Contestants of group D of the National training camp in Informatics love to play a modification of the popular "game of Josef". In the original game, the students are aligned in a ring and, starting from arbitrary chosen student, are labeled with the numbers from 1 through N in clockwise direction. In such a way, the student next in clockwise direction, after the student with the label N is the student with the label 1. The tutor of the group chooses a positive integer K. Starting from the student with label 1, the tutor counts K students and the K-th student leaves the ring. The game continues with a new counting – starting with the student which was next to the student that leaved the game. The K-th student in this counting leaves the game again, and so on, till the moment when only one student remains in the ring – she/he is the winner of the game. In the modification each student chooses her/his own number K_i. First counting is done with the number K of the tutor. Each one of the following countings is done with the number of the student that leaved the game after the previous counting. Write a program **`ring`** to find the label of the student, which will win the modified game.

Input

The first line of the standard input contains the numbers N and K ($N \leq 1\,000\,000$, $2 < K \leq 100$), separated by a single space. The i-th of the next N lines contains the integer K_i – the number chosen by the student with label i ($2 < K_i \leq 100$).

Output

On the single line of the standard output, your program has to print the label of the student, which will remain the last in the ring and will win the game.

EXAMPLE

Input	Output
5 3 3 7 4 3 5	1

Round 2 / Task D2. PETS

Each home pet is identified by a code, composed of capital Latin letters. For example, the code of the cat is `CAT` and the code of the guinea pig is `GPI`. Given a code of some pet, we have to decide to what it is more similar – to the cat or to the guinea pig. The likelihood of a pet *X* with the cat is measured by the number of letters `C`, `A` and `T` in the code of *X*. The likelihood of a pet *Y* with the guinea pig is measured by the number of letters `G`, `P` and `I` in the code of *Y*. Write program **`pets`**, which for given codes of home pets has to decide to which pet any of the given pets is more similar – to the cat or to the guinea pig.

Input

On the first line of the standard input, an integer *L* is given ($2 \le L \le 5$). On each of the next *L* lines, a code of one pet is given – a sequence of capital Latin letters, no longer than 20 characters.

Output

For each of the given *L* pets, your program has to print on a separate line of the standard output one of the following: `CAT`, if the pet is more similar to the cat than to the guinea pig; `GPI`, if the pet is more similar to the guinea pig than to the cat; `CAT-GPI`, if the pet is equally similar to both the guinea pig and the cat. If the code of the pet has no common letters neither with the code of the cat or with the code of the guinea pig, your program has to print `UFO`.

EXAMPLE

Input	**Output**
`4`	`CAT`
`KOTARAK`	`GPI`
`HIPOPOTAM`	`CAT-GPI`
`PAPAGAL`	`UFO`
`ZMEY`	

Round 2 / Task D3. NUMBERS

During a lesson of mathematics, Kate drew on the black board a table, part of which is shown in the left hand side of the Figure. For each row of the table, she filled first and second cell with two integers ***a*** and ***b*** between –1000 and 1000, not equal to zero, ***a*** ≥ ***b.*** Then, she calculated values a^2b and ab^2, and filled them in the third and fourth cells of the row.

a	***b***	a^2b	ab^2
12	2	288	48
-1	-1	-1	-1
...	...	...	...
9	-3	-243	81

a	***b***	a^2b	ab^2
12	0	288	0
0	-1	-1	-1
...	...	...	...
0	0	0	81

Figure

Johnny replaced by zeroes some numbers in the table (see the table, right hand side on the Figure). Kate needs a program **numbers**, which for given numbers in a row of the table, tries to replace back zero values by non-zeros in such way, that all four numbers in a row have to be a, b, a^2b and ab^2, respectively. If more than one solution exists, your program has to find the solution with the minimum value of a. If there are many solutions with the same a, then your program has to find the solution with the minimum value of b.

Input

On the single line of the standard input, separated by single spaces, four integers are given – the values of the numbers in a row, modified by Johnny.

Output

On a single line of the standard output, your program has to print the calculated values of a, b, a^2b and ab^2.

EXAMPLE 1

Input	**Output**
`12 0 288 0`	`12 2 288 48`

EXAMPLE 2

Input	**Output**
`0 0 0 81`	`1 -9 -9 81`

Round 2 / Task E1. LILIES

Lillian wished to plant lilies. She bought several bulbs to grow lilies of different colors. The bulbs appeared so identical that Lillian mixed them up. Which is the minimum number of bulbs that Lillian has to plant so that there will be all desired colors among the grown lilies?

Write program **lilies** to compute this number. We assume that each bulb will grow into a plant.

Input
On the first line of the standard input, the number of desired colors is given. On the second line, the number of bulbs for each color, that Lillian had bought, is given.

Output
The number found by your program.

Constraints
All possible colors are less then 8. The number of bulbs of each color is less than 21.

EXAMPLE

Input
```
3
9 6 8
```

Output
```
18
```

Round 2 / Task E2. BARREL

In a tavern, a barrel contained wine, but was not entirely full. The tavern-keeper sometimes added some vine using a top full jug of wine, and also, sometimes he poured the wine into the same, but empty jug, making the jug top full. Each time adding vine, he marked on a barrel a sing '+', and each time pouring vine out, he marked a sign '–'. The barrel was never overfilled and was never emptied.
Write program **`varel`** to compute the remaining volume of wine in the barrel.

Input
On the first line of the standard input, the initial volume of wine in the barrel is given. On the second line, jug's volume is written. On the next line, the number of sings '+' or '–', is given, followed by sings themselves, each on a separate line.

Output
An integer, equal to the remaining volume, has to be written on the standard output.

Constraints
Initial volume is a positive integer, no greater than 1000. Jug's volume is no greater then 20. The total number of signs '+' and '–', is no greater than 50.

EXAMPLE

Input
```
100
10
5
+
–
+
+
–
```

Output
```
110
```

Round 2 / Task E3. DIFFERENCE

Every natural number can be presented as a product of two natural numbers. Sometimes, several different presentations might be possible for the same natural number and also, in some presentations, both numbers that form the product, might be equals.

E.g. $7 = 1 \cdot 7$, $9 = 1 \cdot 9 = 3 \cdot 3$, $12 = 1 \cdot 12 = 2 \cdot 6 = 3 \cdot 4$.

Write program **`dif`**, that inputs natural number N (which is no greater than 1 000 000) and finds its presentation as a product of two natural numbers, that have the minimal difference.

Input
On a line in the standard input, natural number *N* is written as a positive integer.

Output
Your program has to output both numbers of the product in an increasing order on a line in the standard output, separated them by a space.

EXAMPLES

Input	**Input**	**Input**
`5`	`25`	`48`
Output	**Output**	**Output**
`1 5`	`5 5`	`6 8`

Round 3

Round 3 / Task A1 / B1. GRAPH DIAMETER

An undirected graph G with n vertices and $n-1$ edges is given. The graph is connected and every edge has an integer non-negative length. Denote by $d(x,y)$ the length of the shortest path between two vertices x and y in G. The diameter of the graph G is defined as the largest number $d(x,y)$, where x and y are two arbitrary vertices of the graph. Write a program **`diam`** which computes the diameter of a graph.

Input
The program has to read data from the standard input. On the first line, number n is given ($0 < n < 1000$). The vertices are enumerated by integers from 1 through n. Each of the successive $n-1$ rows describes an edge: the first two numbers are end points of the edge and the third one is its length. The length of any edge is a non-negative integer less than 1000.

Output
The only line of the standard output should contain the diameter of the graph.

EXAMPLE

Input	**Output**

`10`
`4 5 5`
`4 3 2`
`4 2 1`
`5 6 4`
`5 1 0`
`5 7 4`
`3 8 4`
`3 9 3`
`3 10 3`
	`15`

Round 3 / Task A2 / B2. POLYGONS

Given are N different points in the plane. No any 3 of them are collinear. Write program **`poly`** that finds out the smallest area of a convex polygon with K vertices which are taken from among the given points.

Input
Two integers, N and K, are written on the first line in the standard input. It follows N lines, each containing a pair of coordinates for the corresponding given point. Every two numbers on a line in the input are separated by a space.

Output
Your program has to output an integer that is equal to the integer part of minimal area. If there does not exist any convex polygon as is described above, your program has to output 0.

Constraints: $0 < N < 50$, $0 < K < 11$. The coordinates of the given points are nonnegative integers, less than 9999.

EXAMPLE

Input
```
4 3
0 0
1 1
0 10
10 0
```

Output
```
5
```

Round 3 / Task A3 / B3. COLORING

A rectangular table with n rows and k columns is given. Two cells are called adjacent when they have a common side. Write program **`color`** which calculates the number of different ways for coloring the table in such a way that each cell is black or white, and it has exactly one adjacent black cell.

Input
The program has to read data from the standard input. In a single line, the values of n and k are given ($0 < n < 500$, $0 < k < 32$).

Output
The number of different colorings should be written as an integer on a line in the standard output.

EXAMPLE

Input
```
2 2
```

Output
```
4
```

Round 3 / Task A4 / B4. TABLE

Consider 12 different positive integers. They are said to be "correctly" arranged in a rectangular 3 x 4 table (three rows, four columns) if the sum of the numbers in each row is even, and the sum of the numbers in each column is divisible by three. This is maybe attainable in many ways. We call two arrangements "equivalent" if one of them is achieved from the other by changing the places of some rows and/or some columns. It is easy to see that changes of this kind preserve the desired properties for rows and columns, i.e. the correctness of the table. The first arrangement shown below, for example, meets the

conditions for rows and columns, so it is correct. The second arrangement is equivalent to the first one – it is produced by rearranging rows (in fact, by swapping the previous first and second rows, while leaving the third unchanged). Arrangement 3 below is produced by rearranging columns in arrangement 2. All thee arrangements shown are equivalent one to the other.

3	4	29	14
5	10	36	33
7	37	31	13

Arrangement 1

5	10	36	33
3	4	29	14
7	37	31	13

Arrangement 2

33	36	5	10
14	29	3	4
13	31	7	37

Arrangement 3

Write program **`table`** to determine in how many non-equivalent ways a correct arrangement can be constructed out of given numbers.

Input
Two lines are read from the standard input, each of them containing 12 space-separated different positive integers. None of the numbers has more than 50 decimal digits.

Output
For each of the input lines, your program should write on the standard output a corresponding line, containing the number of non-equivalent correct arrangements in a 3 x 4 table of its twelve integers.

EXAMPLE

Input
```
6 3 1 7 9 11 12 10 2 4 5 8
6 12 18 24 30 36 42 48 54 59 60 61
```

Output
```
40320
0
```

Note: In 40% of the test data the input numbers do not exceed 100.

Round 3 / Task A5 / B5. MATRYOSHKA DOLLS

After wining every informatics competition in the world, Peter the Hacker decided to try a different type of challenge. He registered himself in a Russian web site for online competitions in cryptography and locksmith “VerhShifr” and soon he has arrived in Moscow, participating in the finals.

During one of the excursions that the organizers prepared, the group visited a large museum for Matryoshka dolls – traditional Russian nested toys. Inside the museum, there was a strange labyrinth. Its rooms and the labyrinth itself were squares and their walls were parallel to the north-south and east-west axes. Each chamber had four doors – one on each side. The entrance to the labyrinth was in the most northwestern room and the exit – in the most southeastern.

The Russian national team in ciphers prepared an interesting game for their opponents. In each room, they left a single empty Matryoshka doll with unique size. Everyone had to go

through the labyrinth and collect as many toys as possible but under one condition. Every time someone picks a doll, he has to nest all the dolls he has into the new one. Of course, the contestants could decide whether to pick a Matryoshka (if it is bigger than the last one) or to continue to the next room. To make things even harder, the genius of the safes – Metr Pitrichev – had modified the doors so that from a room one can move only to the rooms to the south and to the east.

Peter the Hacker, of course, was not such a master with the locks and could not deal with the modification. Instead, he wrote a program, which can find the best way to collect as many toys as possible. Can you do that, too? Write a program **`matr`** that calculates the maximum amount of Matryoshka dolls, which Peter the Hacker can collect for a given configuration of rooms.

Input
On the standard input you will get a single integer N ($1<N<1000$), followed by N rows with N numbers (one for each room). Every number describes the size of the toy inside the corresponding chamber. The rows are ordered from north to south (the first row is the most northern) and the columns are ordered from west to east (the first number on every row is the most western). A Matryoshka doll fits into another if it has smaller size than the second one. The sizes will be integer numbers in $[1, N^2]$ and will not repeat.

Output
On the standard output, you should print a single integer – the maximum amount of toys that Peter the Hacker can collect following the rules.

EXAMPLES

Input	**Input**
3 9 4 1 6 5 7 2 3 8	4 1 15 4 5 2 10 9 16 14 7 8 3 13 6 11 12
Output	**Output**
4	6

Round 3 / Task A6 / B6. SYMMETRIC POLYNOMIALS

Given is a positive integer N. A polynomial of N variables $x_1, x_2, \ldots, x_N$ is called symmetric, if it is not changed when any permutation of variables' indices is applied. As an example, at N=3, each of the polynomials $x_1+x_2+x_3$, $x_1x_2+x_2x_3+x_1x_3$ and $3x_1x_2+2x_2x_3+3x_1x_3+x_2x_3$ is symmetric, while x_1x_2 is symmetric at N=2, and it is not symmetric at N=3. Write program **`sympol`** that determines if a given polynomial is symmetric.

Input
On the first line in the standard input, the number K of polynomials, that your program has to check, is written. It follows descriptions of these K polynomials. The data for each polynomial start with a line, containing values of N and M, the number of variables and the number of terms of the polynomial. In the next M lines, the polynomial is described. We assume the

polynomial is represented as a sum of terms, each term being a product of a coefficient and variables, each variable raised to a given power. Accordingly, each of the next M lines corresponds to a term and contains a coefficient, and the values of powers to the variables x_1, x_2, …, x_N. Numbers in every line in the input are separated by spaces.

Output
On a line in the standard output, your program has to write a string of zeros and ones (digits 0 and 1, without any spaces). The digit 1 has to correspond to the case when the consecutive polynomial has the symmetric property, and the digit 0 has to correspond to the case, when it has not this property.

Constraints: $0 < K < 21$, $0 < N < 20$, $0 < M < 9999$. The coefficients of each term are nonzero integers in a range from –99 through 99. The values of powers in the terms are nonnegative integers less than 5.

EXAMPLE

Input
```
2
3 1
1 1 1 0
3 4
3 1 1 0
2 0 1 1
3 1 0 1
1 0 1 1
```

Output
```
01
```

Round 3 / Task C1. GAME

A game, like chess is played on a square board, divided into N^2 cells. The cells are numbered with integers from 1 through N^2, row by row (Fig. 1). The cells are white and black, but are colored in a non-regular way (Fig. 2).

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Fig. 1

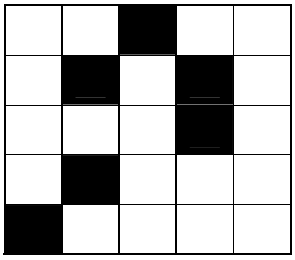

Fig. 2

A pawn is moving only on the white cells. It is possible to move the pawn from a cell to another cell, only if both cells are on the same horizontal or vertical line, and there is no black

cell between them. For example, for the coloring shown on Fig. 2, it is possible to go from cell 13 to the cells 8, 11, 12, 18, 23, in one move.

Write program **move**, which on given the size and the coloring of the board, finds the minimum number of moves needed for moving the pawn from one cell to another.

Input
On the first line of the standard input, three integers are given *N*, *X*, *Y*, representing correspondingly the size of the board and the numbers of the starting and the ending cell. On the second line, the number *B* of the black cells is given first, followed by *B* numbers of black cells themselves.

Output
On a line of the standard output, the program should output the minimum number of moves for moving the pawn from cell *X* to cell *Y*. If it is impossible to move the pawn from cell *X* to cell Y, the program should output –1.

Constraints: $0 < N < 1000$

EXAMPLE

Input
```
5 2 4
6 3 7 9 14 17 21
```

Output
```
7
```

Round 3 / Task C2. TRANSFORMATIONS

Let *a* and *b* be two positive integers with equal number of digits, each digit is not 0. At one move we can change *k* successive digits in *a*, replacing digits 1, 2, 3, 4, 5, 6, 7, 8 and 9 with 2, 3, 4, 5, 6, 7, 8, 9 and 1, respectively. For example, if $a = 12349$ and $k = 2$, at one move we can derive from *a* the following numbers: 23349, 13449, 12459 or 12351. Write program **trans** which calculates the minimum number of moves by which number *b* can be derived from number *a*.

Input
The program has to read the input data from the standard input. The value of *k* ($1 < k < 10$) is given in the first line; the numbers *a* and *b* ($a \neq b$), each of them having at least *k* and at most 100 digits, are given in the next two lines.

Output
The only line of the standard output should contain the calculated number of moves. If *b* cannot be derived from *a*, the program should output 0.

EXAMPLE

Input
```
2
123456789
123467791
```

Output
```
2
```

Round 3 / Task C3. STRANGE WORDS

Distracto likes to invent various strange words instead of learning his lessons. His last idea was to make such words so that if one letter occurs once – it should occur just once more. That way, each letter from the word should be occurred even times. Except that all the letters that are found between the two occurrences of the letter should form word that has the same property. If between the two occurrences of the letter there are no other letters – the word is considered strange. For example the words "abbacc", "aaabbbba", "dddd" are strange according Distracto's conditions, but the words "abcbac", "aaa", "aaabbbab" – aren't.

From this moment forward, Distracto's occupation was to check each word whether it fits his "strangeness" (if it fits to the conditions). Something more, he wants to know the longest strange word in each lesson. This checking takes him so much time, so his marks at school go down to F's only. Help him by making program **`sword`**, which reads text from the standard input. The text is composed by words, separated by single spaces, and written by Latin letters only. Your program should not be capital and small letters case sensitive and should print out the longest word from the input text.

Input

From the standard input your program should read text, composed by words, written in small and capital Latin letters. The words are separated by single spaces. It is sure, that the text contains at least one strange word.

Output

Your program should print a single line on the standard output with the longest strange word, found in the text. The word should be printed in small letters only. If there is more than one such word, your program should print that one, which occurs first in the text.

Constrains

The text may contain no more than 20000 words, each of which can have no more than 100 letters.

EXAMPLE

Input
```
abbacc dd ne da alabala soos sos abcbacb
```

Output
```
abbacc
```

Round 3 / Task C4. POLYGON

A polygon with sides, parallel to the coordinate axes is specified by its horizontal sides only. Write program **`poly`** that computes the area of the given polygon.

Input
The first line of the standard input contains an integer N ($2 \le N \le 1000$) – number of the horizontal sides. Each of the next N lines describes one horizontal side and contains three integers – abscissas of the left and of the right side ends and their ordinate ($-1000 \le$ abscissas, ordinates ≤ 1000).

Output
The only line on the standard output contains a single integer – the area of the polygon.

EXAMPLE

Input
```
3
0 10 10
10 20 0
0 20 20
```

Output
```
300
```

Round 3 / Task C5. SCALE

Given a balance scale with a set of masses: 1, 3, 9, 27, …, 3^{n-1} kg. We have only one copy of each mass. An object with weight of m kg is put on the left scalepan. Write a program **`scale`** that finds a proper distribution of the given masses on both scalepans, so that the balance is reached. It is not necessary to use all the masses.

Input
Your program reads the only line of the standard input with two integers – n ($1 \le n \le 20$) and m ($1 \le m \le 3^{n-1}$).

Output
Your program should write two lines on the standard output. The first line contains integer m, followed by the masses placed on the left hand scalepan in an increasing order. The second line contains the masses placed on the right hand scalepan, also in an increasing order.

EXAMPLE

Input
```
4 5
```

Output
```
5 1 3
9
```

Round 3 / Task C6. DANCE

In a dance club, a dancer is moving backwards to the special spot. The spot is located at x paces to the left and at y paces to the back from the initial position of the dancer. The dancer uses a sequence of two types of dancing steps. With the first one, he moves two paces to the left, and with the second – two paces to the back. Write program **`dance`** that computes the number of different ways in which the dancer can reach the spot.

Input
Enter x and y from the first line of the standard input ($1 < x, y < 61$).

Output
The output shuld contain an integer, equal to the number of variants for which the dancer can reach the spot. If the spot cannot be reached in the described way, the output should be 0.

EXAMPLE

Input
```
4 4
```

Output
```
6
```

Round 3 / Task D1. EXPRESSION

A correct arithmetic expression is given. It contains only sings '+' and '*', and one-digit positive integers ('+' stands for addition, '*' for multiplication). Signs '+' and '*' alternatively appear in the expression forming the sequence '+', '*', '+', '*', ... An example of such an expression is 1+2*3+4*5. The total count of signs '+' and '*' does not exceed 10.

Write program **`expres`** that computes the value of the expression (according to the well-known arithmetic rules).

Input: Your program has to read the expression by the standard input as a string.

Output: Your program should output the result as an integer on the standard output.

EXAMPLE

Input
```
1+2*3+4*5
```

Output
```
27
```

Round 3 / Task D2. TREASURE

After a heavy struggle for their lives at deep sea in bad stormy weather, shipwreck survivors have got into an uninhabitable island. There they found an unusual message, which was written on paper and was placed in a bottle. On the first line of the paper, there was written that a big treasure was hidden somewhere in the island and you could find this treasure if you are able to read the message. The rest of the message contained *N* lines of nonsense text with capital and small letters. In order to encode the message, you have to choose a letter for each line and build a sentence using these letters. The rule to determine which letter to choose is to search for the smallest letter which does not appear in the line and which is greater than the smallest letter occurred in the line (the arrangement of letters is assumed according to the alphabetical order: one letter is less than another, if it is placed before the second one in the alphabet; also, any capital letter is placed before any small letter). If the above defined letter does not exist (for example in case the line contains all the capital and small letters of the alphabet, or the line contains only letters 'z', etc) we assume that we have to take sign '.', instead of a letter. Write program **`treasure`** that tells the survivors what is the message.

Input
On the first input line, the value of N ($1 \le N \le 10000$) is given. On each of the next N lines, a sequence of up to 10000 characters is written.

Output
Your program has to write the message on the standard output as a sequence of N characters.

EXAMPLE

Input
```
7
SPhA
lgfhpqv
efys
znSZww
pnqo
cdbay
psxd
```

Output
```
BigTree
```

Explanation: The letter `A` is the smallest by alphabetical order in the first line. The letter `B` comes after it and `B` does not occur in the same line. The letter `f` is the smallest one in the second line. The letter `i` is the first one that comes after it by alphabetical order and that does not occur in the same line, because `g` and `h` occur.

Round 3 / Task D3. DIGIT

Johnny has started entering in a file one by one numbers 1, 2, 3, …, 9, 10, 11, … Write program **`digit`** that determines which is the *N*-th entered digit.

The value for N is entered by the standard input ($1 \le N \le 10000$). The digit, found by your program, has to be printed on the standard output.

EXAMPLES

Input	**Input**	**Input**
`1`	`11`	`15`
Output	**Output**	**Output**
`1`	`0`	`2`

Round 3 / Task E1. RHOMBS

Let us consider a figure, containing three rhombs, which are inscribed one into another and having their sides formed by asterisks (`*`) :

```
  *
 ***
*****
*** ***
*****
 ***
  *
```

Two asterisks form a side of the innermost rhomb; the side of every next rhomb contains an asterisk more. Write program **`fig`** that, for given N, outputs similar figure formed by N inscribed rhombs.

Input
On the first line of the standard input, the number N is given.

Output
The figure has to be printed on the standard output. No empty lines should appear before the first line that contains an asterisk. No spaces should appear before the leftmost asterisk of the figure.

Constraints: $0 < N < 15$

EXAMPLE

Input
`3`

Output

```
    *
   ***
  *****
*** ***
  *****
   ***
    *
```

Round 3 / Task E2. ROUND TRIP

Johnny with his bicycle started a round trip in Bulgaria. Every day he wrote down the travel time for the day, expressed it in hours and minutes, and the value in kilometers for the day distance.

Write program **`cycling`** that outputs the total travel time for the whole trip, expressed it in days, hours and minutes, and also outputs which day Johnny drove with the maximal velocity (the velocity each day is equal to the value in meters divided by the value in minutes for the travel time this day).

Input
Your program has to read by the first line from the standard input number N of days ($0 < N < 10$). Each of the following N lines contains 3 integers, separated by spaces and these integers are equal to the hours and minutes for the travel time this day, and the distance in kilometers for the day.

Output
On a single line in the standard output, your program should write 4 integers, separating each consecuitive two of them by a single space. The values of the first 3 integers should be equal to the total traveled time in days, hours and minutes, and the last integer should be equal to the position number of the day, during which the velocity is the greatest.

EXAMPLE

Input

```
3
20 15 400
10 30 320
5 0 40
```

Output

```
1 11 45 2
```

Round 3 / Task E3. IRREDUCIBLE FRACTION

Write program **`frac`**, which reduces common fractions.

Input
On the first line of the standard input, two integer a and b are given, the numerator and the denominator of the fraction, respectively, $0 < a < b < 1000$.

Output
On a single line in the standard output, your program should output the numerator and the denominator of the fraction after all possible reductions were done.

EXAMPLES

Input	**Input**	**Input**
`4 10`	`22 35`	`315 385`
Output	**Output**	**Output**
`2 5`	`22 35`	`9 11`

Authors of the problems

Boyko Bantchev:	Round 2 / A2
Veneta Bogdanova	Round 2 / D2
Zornica Dzhenkova:	Round 1 / E3, Round 2 / E1, Round 3 / D2
Katalina Grigorova:	Round 3 / C4, C5
Plamenka Hristova:	Round 3 / E1
Emil Ibrishimov:	Round 3 / A5
Stoyan Kapralov:	Round 3 / A1, C1, D3, E3
Emil Kelevedjiev:	Round 1 / A2, D3; Round 2 / A3, B2, B3, E2, E3; Round 3 / A2, A6, D1, E2
Kinka Kirilova:	Round 2 / D3
Krassimir Manev:	Round 2 / C1, C2, C3, D1
Mladen Manev:	Round 1 / A1, B1, C1, C2, D1, E1, E2; Round 2 / A1, B1; Round 3 / A3, C2
Galina Momcheva:	Round 3 / C6
Pavlin Peev:	Round 1 / A3, B2, B3, C3, D2; Round 3 / A4
Bisserka Yovcheva:	Round 3 / C3