Problems
for
JOI 2005-2006

The 5th Japanese Olympiad in Informatics

The Japanese Committee for the IOI
7-26-37-2D, Shinjuku, Shinjuku-ku, Tokyo
Japan 160-0022
http://www.ioi-jp.org/

Contents

1 First Round January 15, 2006

General Rules for the First Round (Competition on the Website)

1. **First Round on the Web** The First Round of JOI 2005-2006 Competition will be held on the website of JOI, `http://www.ioi-jp.org/`. The website will be open for those who has been made entry for JOI 2006-2007 in advance.

2. **Web Brouser** The Web Competition System on the JOI website has been constructed and tested by using the latest versions of Internet Explorer, Netscape, and some other web brousers. You are recommended to try the System before the start of the competition by using your own PC and a web brouser on it.

3. **Connection Test** From 10:00 on the day before the competition day to 10:00 on the competition day, you can connect to the Web Competition System to try looking at the example problems and/or uploading solutions for them.

4. **Regulations** See the Regulations for Round 1 below.

5. **Logging In** The first thing for you to do is to access the JOI website to log in the Web Competition System. Then, follow the directions described there.

6. **During the Competition** The time shown on the System is just for your convenience which is not necessarily exact.

   There is a possibility to inform the competitors some urgent messages. You are recommended to click the "Urgent Information" button sometimes during the competition.

7. **Problems** You will be given 3 hours (from 13:00 to 16:00) to solve 5 problems. You should write (a) program(s) to solve each problem. The program(s) must output correct answers with five input files, $in1$ to $in5$.

8. **Points** Either five points (for a correct answer) or nothing (for any incorrect answer) is assigned for each input file. Thus, the full score is 100 points.

9. **Solution** You should get a solution for each problem by executing the program(s) you will make by yourself on your own computer.

10. **Submission of Solutions** The solutions you get by executing your program on the given inputs should be submitted through the Web Competition System, following the guidance shown on the system. You are asked to submit the programs you will make as well.

11. **Grading** Grading the examinees will be according to the points they earned.

12. **Questions** If you have any question during the competition, give us e-mail at `joi@ioi-jp.org`.

Regulations for the First Round

Regulations for the First Round

(1) **Programming Languages** You are permitted to use any programming language, except for some powerful symbolic manupulation systems such as *Mathematica*. See term (3) below, too.

(2) **Machines and Operating Systems** You are permitted to use any personal-use computer which is connected to the Internet and equipped with any operating system.

(3) **Multiple Use of Languages** Using multiple programming languages and/or program development environments are prohibited. For example, you are not allowed to use C language for a problem and Java for another.

(4) **# of PCs to Use** Only one PC is allowed to use. Do not use any input device other than a mouse and a keybord.

(5) **User of the PC** During the competition, only a single user (competitor) is allowed to use the PC.

(6) **Online Manuals and Libraries** You are permitted to use online manuals, online helps, and libraries, associated with the programming einvironment on the PC.

(7) **Pre-installation** Do not install any sample program and any helpful information for programming, in advance of the competition.

(8) **Use of Electronic Medias/Devices** Do not use any external medias such as floppy disks and CD-ROMs. Electronic calculators and dictionaries are also prohibited to use.

(9) **Do Not Consult Any Person** During the competition, you should not contact any person with any communication media/tools. Thus, chatting on the Internet as well as communicating by e-mail are prohibited.

(10) **Access to Web Pages** Accessing any website other than the Web Competition System is prohibited during the competition. It is also prohibited to access any other machine on the network by using telnet, ssh, etc.

(11) **Applications Prohibited to Use** Applications other than program development environments, web brousers, and mailers are prohibited to use. For example, do not use any powerful arithmetic manipulation softwares such as *Mathematica, Maple, MuPad, Maxima* and table-calculation softwares such as *Excel*.

(12) **Automatic Program Generating Tools** Do not use any tools which can produce programs, such as *yacc* and *lex*.

1.1 PROBLEM 1

Consider the following card game to be played by Players A and B. Each player is given n cards. A number between 0 and 9 is printed on the right side of each card. Before starting a game, each player lays their n cards wrong side up on their own line, a line for each player. A game consists of n turns. In the first turn, both A and B pick the leftmost card up on their own line. If the number printed on the card picked up by a player, say A, is larger than B's, then A wins to gain the two cards (i.e., A's card as well as B's). If the numbers printed on the two cards are the same, then both A and B gain their own card. Thus at the end of the first turn, there remain $n - 1$ cards on each player's line of cards. The second turn will be played similarly for the remaining cards. Repeating the turns n times, no cards remain on each line of the players, and the game is over.

The score of a player is the sum of numbers printed on the cards gained by the player.

Your task is to write a program that simulates a game, that is, it reads and lays n cards for each player, executes a game with the cards, and outputs the score of each player, where $0 < n \leq 10000$.

INPUT

The first line of each input file contains an integer n, the number of cards given to each player. The $i + 1$-st line ($i = 1, 2, \cdots, n$) contains 2 integers separated by a single space character: the first one is the number printed on the i-th card (counting from left to right on the line) of A, and the second one is the number printed on the i-th card of B.

OUTPUT

Your program should output a line which contains the scores of A and B in this order, respectively, separated by a single space character. The line should end with the Return code.

EXAMPLE

Example input 1	Example input 2	Example input 3
3	3	3
9 1	9 1	9 1
5 4	5 4	5 5
0 8	1 0	1 8

Example output 1	Example output 2	Example output 3
19 8	20 0	15 14

1.2 PROBLEM 2

Write a program to transform a given string of symbols to another, based on a given table of transformation.

Each string consists of Roman letters and digits, where uppercase letters and lowercase letters should be distinguished. There is no rule on appearance of letters in the transformation table.

In the transformation table, each line contains 2 letters/digits separated by a single space character, meaning that whenever the first letter appears, it should be transformed into the second one.

The transformation should be done just once for each occurrence of letters and digits. In other words, an occurence of a letter/digit in the original string should not be transformed again even if the transformed letter is one of the letters/digits in the transformation table. Any letter/digit which does not appear in the transforamtion table should be transformed into itself.

INPUT

The first line of each input file contains an integer n, the number of lines of the transformation table. The $i+1$-st line ($1 \leq i \leq n$) contains 2 letters, say p and q, separated by a space character, which means that p should be transformed into q. The $n+2$-nd line contains an integer m ($0 < m < 10^8$), the length of a string to be tarnsformed. The consecutive m lines beginning from the $n+3$-rd line represent each letter/digit of the string, that is, $(n+2+j)$-th line ($1 \leq j \leq m$) contains a letter/digit which is the j-th letter/digit of the string to be transformed.

OUTPUT

The output file should contain a single line consisting of a string obtained from the input string by the transformation, followed by the Return code.

EXAMPLE

Example input 1	Example output 2
3 A a 0 5 5 4 10 A B C 0 1 4 5 a b A	aBC5144aba

1.3 PROBLEM 3

Let us consider a dice placed on a desk as shown in Fig. 1.

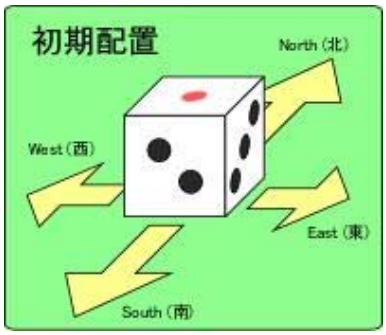

Figure 1: Initial configuration.

As you see, number 1 is on the upper side, 2 on the south side , 3 on the east side. Since the sum of the numbers on each pair of opposite sides is 7, it follows that 5 is on the north side, 4 on the west side, and 6 on the lower side, all of which are not seen in Fig. 1.

Starting with Fig.1 as the initial configuration, rotate the dice obeying given instructions. The instructions are one of 6 operations shown in Figs. 2 and 3.

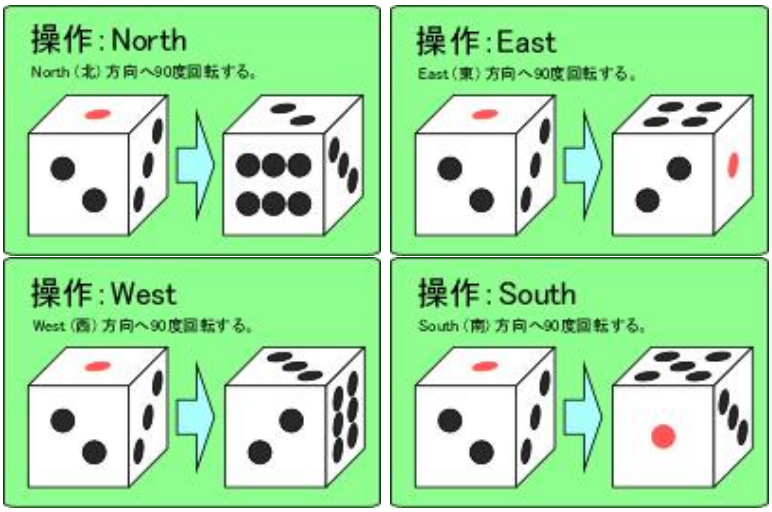

Figure 2: Rotations “North”, “East”, “South”, and “West”.

“North”, “East”, “South”, and “West” are operations to rotate the dice 90° in the designated direction, as shown in Fig. 2.

“Right” and “Left” in Fig. 3 are operations to rotate the dice 90° horizontally in the right and left directions, respectively.

Your task is to write a program to perform operations successively accdording to a given sequence of instructions, starting from the initial configuration shown in Fig. 1. The program should output the sum of the numbers (including ”1” of the initial configuration) appeared on the upper side of the dice in the course of rotations.

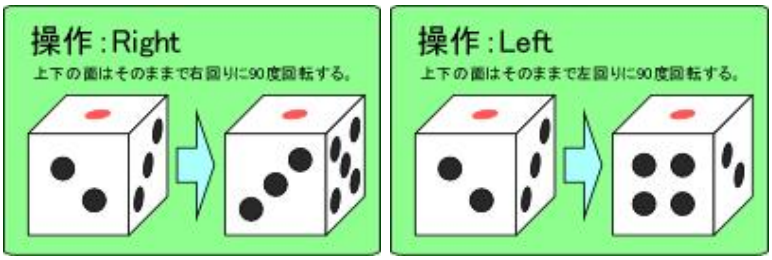

Figure 3: Rotations “Right” and “Left”.

INPUT

The first line of an input file contains an integer n, the number of operations to be performed. The $i+1$-st line contains the i-th operation to perform, one of 6 operations, where $1 \leq i \leq m \leq 1000$.

OUTPUT

Each output file should contain the sum, followed by the Return code at the end of line.

EXAMPLE

Example input 1	Example input 2
5	8
North	West
North	North
East	Left
South	South
West	Right
	North
	Left
	East

Example output 1	Example output 2
21	34

1.4 PROBLEM 4

There are n cups of different sizes, and three trays A, B, and C. Initially the cups are piled up upside down on the trays, at most one pile on each tray, such that the smallest cup (among the cups on the tray) is at the bottom, on which the second smallest, on which the third smallest, and so on.

For example, Fig. 1 shows the case where $n = 5$ and 2 cups are piled up on tray A, nothing on tray B, and 3 on tray C.

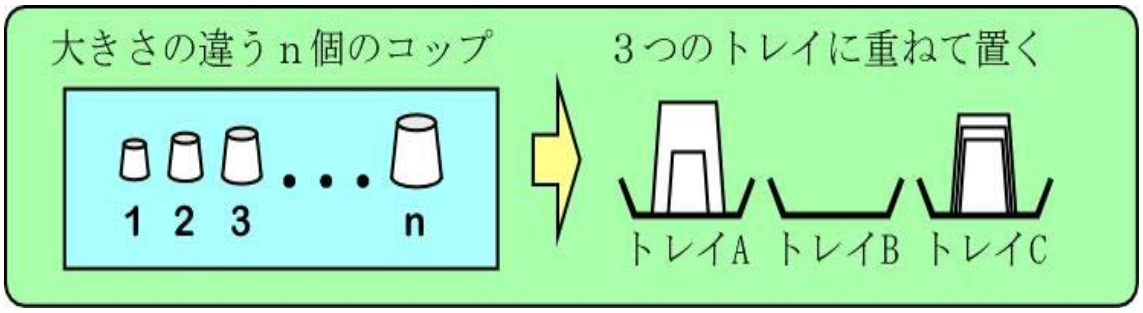

Figure 4: An initial configuration, where $n = 4$.

Given an initial configuration, find the smallest number of steps to move all the cups so that they are piled up either on tray A only or on tray C only, where the moves must obey the following three rules.

Rule 1 At each step, only the topmost cup on some tray (which is the largest one on the tray) can be moved to be piled up on the topmost cup on another tray.

Rule 2 Do not pile up any smaller cup on any other cup.

Rule 3 Neither direct move "from tray A to C" nor "from C to A" is allowed. Thus, a direct move must be one of "from A to B", "from B to A", "from B to C", and "from C to B."

Your task is to write a program which, given an initial configuration of n cups and an integer m, determines whether or not all the cups can be moved to be piled up on tray A whithin m steps. If it is possible, then the program should output the smallest number of steps to reach the final configuration, and -1 otherwise.

INPUT

The first line of each input file contains 2 integers n, the number of cups ($1 \leq n \leq 15$) and m, an upper bound of the number of steps permitted ($1 \leq m \leq 15000000$), separeted by a space character. Each of the 2nd, 3rd, and 4th lines contains an integer k, followed by a sequence of integers $c_1, \cdots, c_k$ taken from $\{1, 2, \cdots, n\}$ sorted in the ascending order. The k on the 2nd, 3rd, and 4th lines represent the numbers of cups piled up initially on trays A, B, and C, respectively. The $c_1, \cdots, c_k$ on the 2nd, 3rd, and 4th lines represent the (sizes of) cups piled up on trays A, B, and C, respectively.

OUTPUT

Each output file must contain a single line containing either the required number of steps or -1 followed by the Return code.

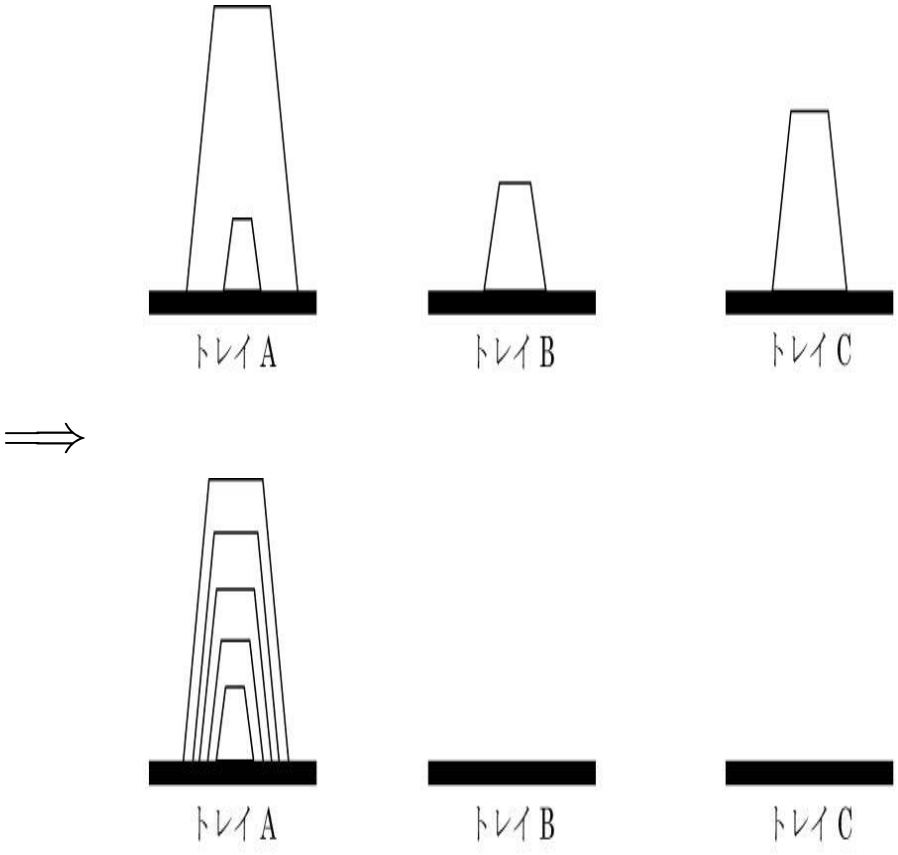

Figure 5: All of the $n\,(=4$ in this case) cups are moved to be piled up on tray A.

EXAMPLE

Example input 1	Example input 2	Example input 3	Example input 4
3 10 0 1 1 2 2 3	4 20 2 1 2 1 3 3	2 5 2 1 2 0 0	3 3 0 1 1 2 2 3
Example output 1	**Example output 2**	**Example output 3**	**Example output 4**
9	3	0	-1

1.5 PROBLEM 5

Consider the following card game. You are given k packs of n cards numbered $1, 2, \cdots, n$. Number i is printed on the right side of the card numbered i. Thus, for each i, there are exactly k cards with number i printed on their right sides. You schuffle the nk cards sufficiently and divide them into n packs, each of which consists of k cards piled up in a certain order. Then, you arrange the n packs in a line, and call the i-th pack from the left on the line the i-th *Mountain*. See Fig. 1.

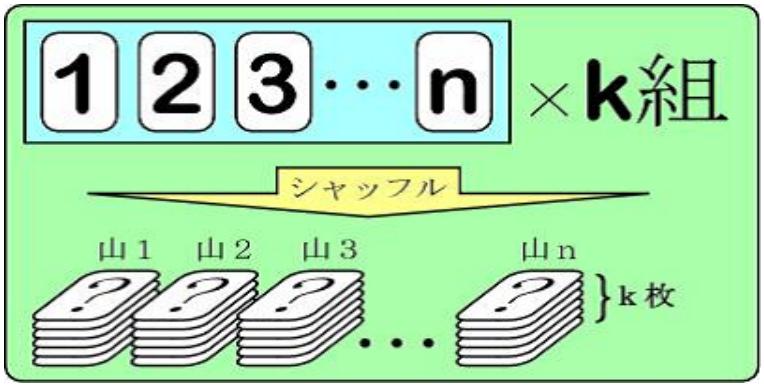

Figure 6: The initial configuration of a game.

You start a game at Mountain 1. Draw the topmost card of the Mountain. Whenever a card is drawn from a Mountain during a game, it will not be put back to the Mountain again. If the number printed on the drawn card is i, then draw the topmost card of Mountain i next time. If, as a result, j is printed on the drawn card, then draw the topmost card of Mountain j next time, and so on. Repeat such process of drawing a card from the designated Mountain, until no card remains in the mountain from which you have to draw its topmost card at that time. Then, you win the game if every Mountain is empty (i.e., no card remains in any Mountain), otherwise you lose. This ends the *first stage* of a game.

Whenever you lose a game, you may continue the game as the *second stage*, as follows. Only those Mountains with at least one remaining cards will be used in the second stage. The numbering for such Mountains remain unchanged. You start the second stage by drawing the topmost card from the leftmost Mountain among the remaining Mountains. Continue the game just in the same way as in the first stage. Like the first stage, if no card remains on the Mountain you have to draw a card next, then the second stage is over. You win in case every Mountain is empty, and you lose otherwise.

For each game, you will be given an integer m. You have to go to the second stage if m is 1 and you have lost the first stage, otherwise the game is over just after the first stage.

Initial configurations for games differ depending on shuffling the cards. There are three possibilities for each game: you win in the first stage, you lose in the first stage and win in the second stage, or you lose both in the first and second stages, depending on the initial configuration. We assume that each possible initial configuration appears with the same probability. Under the assumption we like to find the probability p to win a game.

Your task is to write a program to compute and output the probability p, satisfying the following 2 conditions.

1. p should be written in decimal form with exactly r digits under the decimal point. If $p \times 10^K$ is an integer for a sufficiently large positive integer K, then the digits under the

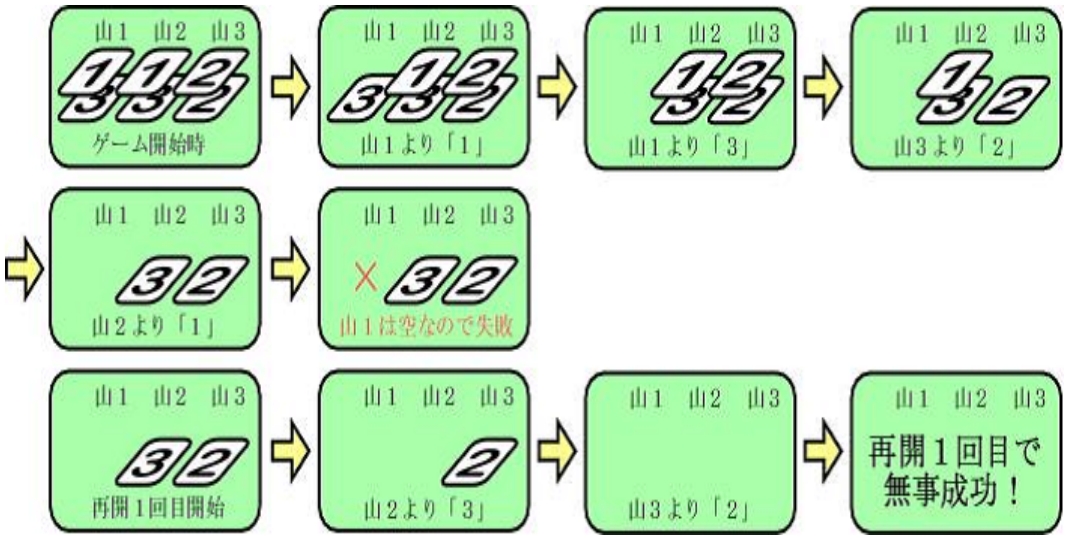

Figure 7: A sequence of moves in a game.

decimal point contain $000\cdots$ at the tail. In this case, the output should contain $00\cdots00$ at the tail so that the number of digits after the decimal point is exactly r. For example, $p = \frac{3}{8} = 0.375$ should be written as 0.37500 if $r = 5$, and 0.37 if $r = 2$. Similarly, if $p = 1.0$ and $r = 3$, then p should be written as 1.000.

2. For example, $0.150000\cdots$ can also be represented as $0.1499999\cdots$. In this case, $0.15000\cdots$ should be used.

INPUT

The first line of any input file contains integers n, k, m and r in this order, separated by a space, where $1 \leq n \leq 10000$, $1 \leq k \leq 100$, m is either 0 or 1, and $1 \leq r \leq 10000$.

OUTPUT

Each output file should contain a single line containing the probability p followed by the Return code.

EXAMPLE

Example input 1	Example input 2	Example input 3
2 1 0 5	3 1 1 3	2 2 1 3
Example output 1	**Example output 2**	**Example output 3**
0.50000	0.833	1.000

2 Final Round February 12, 2006

2.1 Directions for Competitors

1. Do not open the problem book and wait for the invigilator's instruction.
2. There are five problems to solve. The competition will start at 12:30 and end at 15:30.
3. Read the Overview Sheet, where instructions about problems such as the limited amounts of time and memory for each problem are described, as well as the way of how to submit your solutions.
4. The first thing you have to do after the competition starts is to execute file `joi2006.exe` placed in the home directory of *cygwin*:

   /home/[your code] (C:/cygwin/home[your code])

   The password is `JOIioj`.
5. As a result of the above execution, a directory named `joi2006` will be produced as a subdirectory of the home directory. Directory `joi2006` has five subdirectories corresponding to Problems 1 to 5: `2006-ho-t1`, `2006-ho-t2`, `2006-ho-t3`, `2006-ho-t4`, and `2006-ho-t5`. These subdirectories contain sample input data files and their output files.

2.2 Task Overview Sheet

See the next page.

About Tasks ／ Final Round

Task		Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
CPU Time Limit		1 sec CPU	3 sec CPU	1 sec CPU	60 sec CPU	60 sec CPU
Memory Limit		16 MB	16 MB	128 MB	128 MB	128 MB
Compiler Options	C(gcc)	`-Wall -O2 -lm`				
	C++(g++)	`-Wall -O2 -lm`				
	JAVA	`-O`				
# of Input Data		5	5	5	10	10
Points per Input Data		4	4	4	2	2
Total Sum of Points		20	20	20	20	20

Comments at the Top of Programs	C Language	`/*`
		`TASKNO: 1`
		`LANG: C`
		`*/`
			`/*`
			`TASKNO: 2`
			`LANG: C`
			`*/`
				`/*`
				`TASKNO: 3`
				`LANG: C`
				`*/`
					`/*`
					`TASKNO: 4`
					`LANG: C`
					`*/`
						`/*`
						`TASKNO: 5`
						`LANG: C`
						`*/`

	C++	`/*`
		`TASKNO: 1`
		`LANG: C++`
		`*/`
			`/*`
			`TASKNO: 2`
			`LANG: C++`
			`*/`
				`/*`
				`TASKNO: 3`
				`LANG: C++`
				`*/`
					`/*`
					`TASKNO: 4`
					`LANG: C++`
					`*/`
						`/*`
						`TASKNO: 5`
						`LANG: C++`
						`*/`

	JAVA	`/*`
		`TASKNO: 1`
		`LANG: JAVA`
		`*/`
			`/*`
			`TASKNO: 2`
			`LANG: JAVA`
			`*/`
				`/*`
				`TASKNO: 3`
				`LANG: JAVA`
				`*/`
					`/*`
					`TASKNO: 4`
					`LANG: JAVA`
					`*/`
						`/*`
						`TASKNO: 5`
						`LANG: JAVA`
						`*/`

2.3 PROBLEM 1

IOI highschool has a plan to go on a school excursion. To decide where to go, the student council of the school has sent out questionaire to n students numbered 1 to n, where $1 \leq n \leq 1000$.

There are m ($1 \leq m \leq 100$) candidates numbered 1 to m for the destination of the excursion. Students are asked to answer OK to those destinations they like, and NG to others.

Your task is to write a program which compute and output the destinations ordered according to the preference the students made, that is, a destination is to appear before another if the former has gained more OKs than the latter. If some destinations have gained the same number of OKs, then they should be ordered according to their numbers.

INPUT

In your source code, the name of the input file should be `input.txt`. The first line of the file contains two integers, n ($1 \leq n \leq 10000$) and m ($1 \leq m \leq 100$), separated by a single space character. The $i+1$-st line represents the answer of i-th student, where OK is denoted by 1 and NG by 0. Those 1's and 0's should be separated by a space character. The j-th 0/1 is the answer to the j-th destination.

OUTPUT

In your source code, the name of the output file should be `output.txt`. The file should contain a single line containing m integers representing the destinations ordered according to the preference, followed by the Return code.

EXAMPLE

Example Input:

```
4 6
1 0 1 0 1 1
1 1 0 1 0 0
1 1 1 0 0 0
1 0 1 0 1 0
```

Example output:

```
1 3 2 5 4 6
```

2.4 PROBLEM 2

Given a string consisting of decimal digits, $0, 1, \cdots, 9$, such as 122244 and 44444444444, consider the following operation which produces from such a string a new string.

The operation begins with reading the digits of the given string from left to right. If a digit, say a, appears consecutively r times but not $r+1$ times for some positive integer r, then produce $\bar{r}a$ without any space between digits, where $\bar{r}$ is the string of digits representing integer r in decimal notation. Repeat this process on the remainning string (i.e., the substring of the given string beginninig from the $r+1$-st digit) until the remaining string become empty.

By concatenating all the strings produced in this way during the course of processes, without any space between them, a new string of digits is produced. We count the whole course of processes for one application of the operation.

For example, from 122244 a new string $\bar{1}1\bar{3}2\bar{2}4 = 113224$ is produced by one application of the operation, and from 44444444444 a new string 114.

The operation may be applied repeatedly.

Your task is to write a program which, given a string of digits of length less than or equal to 100, outputs a new string that is obtained from the string by applying the operation n times, where $n \leq 20$.

INPUT

The input file is `input.txt` which consists of 2 lines. The first line contains an integer n, the number of times for the operation to be appplied. The second line contains a string to receive the operation.

OUTPUT

The output file should be `output.txt` which should contain a single line containing the new string produced by applying the operation n times, followed by the Return code.

EXAMPLE

Example Input:

```
5
11
```

Example output:

```
13112221
```

2.5 PROBLEM 3

Given n squares of the same size ($n \leq 20$), arrange them on an horizontal line in several columns such that the number of squares in any column should not be less than that of the column on its immediate right.

For example, if $n = 5$, then exactly seven arrangements are possible, as is shown below:

Any arrangement can be represented by a sequence of integers, the number of squares in the columns from left to right. For example, in case $n = 5$ (the case shown in the above figure), they are

$$(5),\ (4,1),\ (3,2),\ (3,1,1),\ (2,2,1),\ (2,1,1,1),\ (1,1,1,1,1).$$

Your task is to write a program which, given n, outputs all possible arrangements in lexicographical order, where lexicographical order is defined as follows: $(a_1, a_2, \cdots, a_s)$ precedes $(b_1, b_2, \cdots, b_t)$, if either $a_1 > b_1$ or there is an integer $i > 1$ such that $a_1 = b_1, \cdots, a_{i-1} = b_{i-1}$ and $a_i > b_i$.

INPUT

The input file is `input.txt` which consists of a single line containing n.

OUTPUT

The output file should be `output.txt` which should contain all the possible arrangements in lexicographical order, an arrangement a line, and ends with the Return code. An arrangement $(a_1, \cdots, a_s)$ should be output as a sequence of integers $a_1, \cdots, a_s$ separated by a single space character between them.

EXAMPLE

Example Input:

```
5
```

Example Output:

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

2.6 PROBLEM 4

Consider n strings with rings at both ends. An integer is attached to each ring such that the integers, say a and b which we denote by $[a, b]$, attached to the both ends of a string are different. These pairs of integers identify the strings.

Two strings $[a, b]$ and $[c, d]$ can be connected if one of a, b is equal to one of c, d, by tieing them together at the rings with the same number. The result is called a *chain*. For example, a chain $[1, 3, 4]$ is obtained by connecting two strings $[1, 3]$ and $[3, 4]$.

Similarly, a string and a chain, or two chains can be connected together at the rings with the same integer. For example, a chain $[1, 3, 4]$ and a string $[5, 1]$ can be connected to produce a chain $[5, 1, 3, 4]$. From two chains $[1, 3, 4]$ and $[2, 3, 5]$, a form looking like a cross (call it α for later reference) can be obtained by tieing them at the center of each string. A form looking like a ring (call it β) can be obtained from two strings $[1, 3, 4]$ and $[4, 6, 1]$ by connecting them at both ends. In this way various forms can be obtained.

A part of such a form is called *chain* if it is a sequence of strings connected at their ends with the property that no two rings with the same integer appear on it. For example, α contains chains $[1, 3, 2]$, $[1, 3, 4]$, $[1, 3, 5]$, $[2, 3, 1]$, $[2, 3, 4]$, etc. of length 3, and β contains chains of length 4 such as $[1, 3, 4, 6]$, $[3, 4, 6, 1]$, $[4, 6, 1, 3]$, where the *length* of a chain is the number of integers on it.

Your task is to write a program to find the maximum lengh of possible chains.

INPUT

The input file is `input.txt`, the first line of which contains an integer n ($1 \leq n \leq 100$), followed by n lines containing two integers separated by a single space character. The $i + 1$-st line ($1 \leq i \leq n$) containing integers a and b ($1 \leq a < b < 100$) represents a string whose ends are rings with integers a and b.

OUTPUT

The output file should be `output.txt` which should contain the maximum length and end with the Return code.

EXAMPLE

Example Inputs:

Input 1	Input 2	Input 3
7	6	7
1 3	1 2	1 3
3 4	2 3	2 4
1 4	3 4	3 5
2 7	4 5	4 6
5 7	1 5	6 7
6 7	2 6	2 6
1 7		4 7

Example Outputs:

Output 1	Output 2	Output 3
5	6	4

2.7 PROBLEM 5

There are many rectangular sheets placed on the plane. Your task is to write a program to calculate the area covered by the sheets, as well as the length of the circumference surrounding the area.

We assume the standard coordinate system to represent the sheets on the plane. The sheets are arranged so that the following conditions are satisfied:

(1) The x- and y- coordinates of four vertices of each sheet are integers between 0 and 10000.
(2) Each side of any sheet is parallel to either the x-axis or the y-axis.
(3) The number of sheets is at most 10000.

INPUT

The input file is `input.txt` and the first line of it contains 2 integers n, the number of sheets, and r, the type of the problem, separated by a single space character. The $i+1$-st line contains four integers x_1, y_1, x_2 and y_2 in this order, separated by a single space character between them, to represent the left-lower coordinate (x_1, y_1) and the right-upper coordinate (x_2, y_2) of the i-th sheet.

OUTPUT

The output file should be `output.txt`. The area should be output on the first line of `output.txt` in case $r = 1$. The length of the cicumference should be output on the second line additionally in case $r = 2$. In each case the last line should end with the Return code.

The coordinates of vertices of sheets are between 0 and 100, 40% among all the input data, and a half of them are restricted to $r = 1$. Also, r is restricted to be 1 for a half among all the input data.

EXAMPLE

Example Inputs:

Input 1	Input 2	Input 3	Input 4
5 1	5 2	2 2	3 2
0 0 3 2	0 0 3 2	0 0 8 9	2 2 8 9
1 1 2 5	1 1 2 5	0 0 9 8	3 0 4 9
0 4 6 5	0 4 6 5		5 0 7 9
3 3 5 6	3 3 5 6		
5 0 7 6	5 0 7 6		

Example Outputs:

Output 1	Output 2	Output 3	Output 4
29	29	80	45
	38	36	36