ACM ICPC 2008–2009 NEERC Moscow Subregional Contest
Moscow, October 26, 2008

List of Problems

Problem	Page	Time limit	Memory limit	Name
A	2	1 second	128 MiB	ACM ICPC Rules
B	3	1 second	128 MiB	Bankrupt Broker
C	4	1 second	128 MiB	Construction
D	5	4 seconds	128 MiB	Database Query Engine
E	7	1 second	128 MiB	Extreme Programming
F	8	1 second	128 MiB	Flight to Mars
G	9	1 second	128 MiB	Genesis Project
H	10	3 seconds	128 MiB	Hadron Trip
I	11	1 second	128 MiB	Mr. Incognito's Real Estate
J	12	4 seconds	128 MiB	Jig-saw Puzzle
K	14	1 second	128 MiB	Klondike Gluon Rush
L	15	1 second	128 MiB	Lucky Bonds

Your solution must read the input data from the standard input and write the results to the standard output. Output to the standard error stream is prohibited.

Unless explicitly stated in the problem statements, all input elements may be separated by an arbitrary number of whitespace characters. All input data are correct and satisfy the specification given in the problem statement.

The output of your program must exactly satisfy the output specification in the problem statement.

Problem A. ACM ICPC Rules

Time limit: 1 second
Memory limit: 128 MiB

The ACM International Collegiate Programming Contest (ACM ICPC) is a multi-tiered competition among teams of students representing institutions of higher education. Teams of Moscow Region first compete in the Moscow Subregional Contest. The winning teams qualify to advance to the Northeastern European Regional Contest to be held in November 28, 2008.

The teams qualifying to the Regional Contest are selected from the winners of the Moscow Subregional Contest according to the following rules. Moscow State University can be represented at the Regional Contest by at most four teams. All other institutions can be represented by at most two teams. School teams do not advance to the Regional Contest. The participating teams are sorted according to their result in the contest. All teams are removed from the list, except the top four teams from Moscow State University and the top two teams from all other institutions. At most the top ten teams in the resulting list advance to the Regional Contest.

Your task is to determine the qualifying teams by the list of the participating teams ordered by their result.

Input

The first input line contains the number of teams N ($1 \le N \le 100$). The next N lines contain team results in the following format:

```
<short institute name><space><team name>
```

where `<short institute name>` is a nonempty sequence of uppercase and lowercase Latin letters of length up to 10. Moscow State University has the short name "`MSU`". For all school teams the short name "`SCH`" is used. `<team name>` is a nonempty sequence of characters of length up to 30. A team name consists of uppercase and lowercase Latin letters, digits, and special symbols '_', '#', '!', '?'.

Output

Output the list of the qualifying teams in the same order and format as specified in the input. The first line of the output must contain the number Q of qualifying teams. Each of the next Q lines must contain the short name of the institution and the team name separated with one space character.

Example

stdin	stdout
13 FBU Fence_Builders_#3 WCU Woodcutters_#1 FBU Fence_Builders_#2 MSU Vasya_Team FBU Fence_Builders_#1 MSU Vanya_Team FBU Fence_Builders_#4 OILU Olive_Oil_Team WWW What?_Where?_When? MSU Vova_Team SCH Pupils_#1 SCH Pupils_#2 WWW Why?_Why?_Why?	9 FBU Fence_Builders_#3 WCU Woodcutters_#1 FBU Fence_Builders_#2 MSU Vasya_Team MSU Vanya_Team OILU Olive_Oil_Team WWW What?_Where?_When? MSU Vova_Team WWW Why?_Why?_Why?

Problem B. Bankrupt Broker

Time limit: 1 second
Memory limit: 128 MiB

After a crash on the stock market one deeply depressed broker had glued his stocks down a lavatory wall. However, after a while he has decided to sell these stocks as quick as possible to pay out interest for his Bentley. He has stocks of many different companies, each identified by an integer number z. The lavatory wall consists of M rows with N stocks in each row.

The broker sells all stocks of some company x in one day. No stocks of other companies are sold in the same day. All stocks of the company x are sold at a fixed price that is equal to the *market index*. In the first day the value of the market index is F_1, the index in the i-th day is $F_i = ((A \cdot F_{i-1} + B) \bmod C) + 1$, where A, B, and C are some integers.

Help the broker to sell all his stocks in the shortest period of time. Between several shortest sequences of sales choose the one with the maximum profit.

Input

The first line of the input contains four integer numbers F_1, A, B, and C $(1 \le F_1, A, B, C \le 10\,000)$. The second line contains two integer numbers M and N $(1 \le M, N \le 250)$. The following M lines contain N company identifiers z $(1 \le z \le 2^{31} - 1)$ each.

Output

The first line of the output must contain the minimal number D of days required to sell all stocks. The second line must contain D company identifiers, whose stocks are to be sold in the corresponding day separated by space. Of several shortest sequences of sales the one with the maximum profit must be chosen. If several solutions exist, output any of them.

Example

stdin	stdout
1 3 1 5 4 4 1 2 3 6 2 1 1 2 3 1 1 3 2 2 2 2	4 6 2 3 1

Problem C. Construction

Time limit: 1 second
Memory limit: 128 MiB

A construction company develops a new office building complex. Each office building is a square of size 100×100 ft in plan. The height of each building is also measured in 100 ft units. The sides of the building complex are oriented along the cardinal directions and have the size of $M \cdot 100$ ft from West to East and $N \cdot 100$ ft from North to South. The office buildings are also oriented along the cardinal directions.

According to a Mayor's directive, the construction company must erect $M \cdot N$ office buildings of the specified heights. However, the choice of location for each building is left to the company. Due to the financial crisis the company cuts the costs and erects buildings without any space between the neighboring ones. Also, the company wants to minimize the area of the outer walls of the complex.

Your task is to determine the location of each office building so that the area of the outer walls is minimized.

Input

The first line of the input contains two integer numbers M and N $(1 \le M, N \le 250)$ specifying the size of the office building complex. The next $M \cdot N$ numbers specify the heights of the building according to the Mayor's directive in the ascending order. The heights are given in 100 ft units. The sum of all heights does not exceed 10^8 ft.

Output

Output the minimum area of the outer walls of the complex measured in 100×100 ft^2 units.

Example

stdin	stdout
2 2 1 2 3 4	26

Problem D. Database Query Engine

Time limit: 4 seconds
Memory limit: 128 MiB

Stock market securities have many properties such as floating price, volatility, liquidity, etc. Also one may estimate their mean profitability, growth potential, etc.

WebMarket is a managing company of a stock exchange. It provides trading facilities for stock brokers and traders, services for issue and redemption of securities as well as other financial instruments and capital events including payment of income and dividends.

WebMarket analysts introduce a new market index called *reliability*. High reliability value implies low risk of buy, but also lower profit estimate.

WebMarket analysts decide to make the reliability index public and provide a service of buying the security with the K-th highest value of reliability. According to WebMarket analysts this instrument will attract more clients and increase the market turnover.

Reliability estimation algorithms are already baked. Your task is to write a program that manipulates the reliability index list. The securities are listed in the decreasing order of reliability. The securities with equal reliabilities are listed in the increasing order of their ids.

Securities in the Webmarket database have three attributes:

- `code` — a non-empty string of Latin letters and digits, less than 30 characters in length;
- `id` — an integer (auto-incrementing key starting from 0);
- `reliability` — an integer in the range $[-2^{31}, 2^{31})$.

For each new security, the value of `id` is automatically assigned to the next available id. Initial value of reliability is 0. If a security is removed its id is never reused.

The following requests should be supported: issuing a new security, removing a security from the database, changing the value of reliability for an existing security, and finding the K-th security in the reliability index list.

Input and Output

The first line contains the number of requests N $(0 \le N \le 100000)$. The next N lines contain requests, one per line. A request has one of the following four forms.

- `ISSUE code`: add a new security `code` or output the security info, if it exists in the database
  - if the security `code` exists, output
    `EXISTS id reliability`
  - if the security `code` does not exist, output
    `CREATED id 0`
- `DELETE code`: remove the security `code` from the database
  - if the security `code` exists, output
    `OK id reliability`
  - if the security `code` does not exist, output
    `BAD REQUEST`

- `RELIABILITY code m`: increase the reliability of the given security by `m`. It is guaranteed that the value of reliability remains in the range $[-2^{31}, 2^{31})$.
  - if the security `code` exists, output
    `OK id new_reliability`
  - if the security `code` does not exist, output
    `BAD REQUEST`
- `FIND n`: find the *n*-th security in the list (starting from zero)
  - if the database is not empty, then output the *n*-th security in the reliability index list; if the list contains less than *n* items, output the last one in the form
    `OK code id reliability`
  - if the database is empty, output
    `EMPTY`

Your program should respond with one line for each request.

Example

stdin	stdout
17 ISSUE aaa FIND 10 ISSUE bbb ISSUE ccc RELIABILITY aaa 10 RELIABILITY bbb 30 RELIABILITY ccc 20 RELIABILITY xxx 20 FIND 1 FIND 2 FIND 0 ISSUE eee ISSUE fff FIND 3 FIND 111 DELETE bbb FIND 0	CREATED 0 0 OK aaa 0 0 CREATED 1 0 CREATED 2 0 OK 0 10 OK 1 30 OK 2 20 BAD REQUEST OK ccc 2 20 OK aaa 0 10 OK bbb 1 30 CREATED 3 0 CREATED 4 0 OK eee 3 0 OK fff 4 0 OK 1 30 OK ccc 2 20

Problem E. Extreme Programming

Time limit: 1 second
Memory limit: 128 MiB

A new "Winners and Losers" series of programming contests has recently gained great popularity among lovers of Extreme Programming. As opposed to the ACM ICPC rules, only one submit per a contestant per a problem is allowed in each contest. The participants are ranked according to the number of problems solved. The participants, who solve the same number of problems are ranked by least penalty. The penalty is calculated as the sum of submit times for all solved problems. Rejected solutions do not affect the penalty.

To define participant's *strategy* one needs to specify a subset of problems and an order in which these problems are to be solved and submitted. The sum of durations required to solve these problems must not exceed the duration of the contest.

A good choice of strategy is a key to success in the contest. So, a strategy is called *optimal* if it maximizes the expectation EP of the number of solved problems. If several strategies have the same expectation EP, the expectation ET of the resulting penalty should be minimized[1].

For each problem you are given the probability of solving it and the duration required to write a program. Your task is to determine the optimal strategy for the given problem set.

Input

The first line contains two integer numbers: N $(1 \le N \le 100)$ — the number of problems, and $T_{\max}$ $(1 \le T_{\max} \le 1\,000)$ — the contest duration. Each of the following N lines contains a pair of integer numbers: T_i $(1 \le T_i \le 1\,000)$ — the duration required to write a solution for the i-th problem, and P_i $(0 \le P_i \le 100)$ — the probability in per cent of solving the i-th problem.

Output

Output the optimal strategy as follows. The first line of the output should contain the number of problems in the optimal strategy. The second line should contain the ordered list of these problems separated by a space.

Example

stdin	stdout
5 150 50 60 70 85 60 60 70 95 40 60	3 5 1 3

[1] Let the problems be ordered according to the chosen strategy. Let k be the number of the problems in the strategy, $T_1, T_2, \ldots, T_k$ denote the durations required to write a solution for these problems, and $P_1, P_2, \ldots, P_k$ denote the corresponding probabilities of success. Then

$$EP = P_1 + P_2 + \ldots + P_k,$$
$$ET = T_1P_1 + (T_1 + T_2)P_2 + \ldots + (T_1 + T_2 + \ldots + T_k)P_k.$$

Problem F. Flight to Mars

Time limit: 1 second
Memory limit: 128 MiB

In October 1, 203* the Only Superpower launches the first manned expedition to Mars on the Avenger spacecraft. To ensure safe landing on Mars, a robotic space probe orbiting Mars has dropped three beacons marking a safe landing area on the Martian surface.

However, to cut the cost of the expedition, Avenger is not equipped with a parachute. So Avenger just moves at a constant speed until the Martian surface is reached.

The taikonaut commander asks you to determine if Avenger hits the surface inside the safe landing area. This area is the spherical triangle with the apexes at the beacon landing points.

Input

The first input line contains three integer numbers (x_0, y_0, z_0) — the Cartesian coordinates of the Martian center. The next input line contains one integer number R $(1 \le R \le 10000)$ — the radius of Mars. The next three lines contain spherical coordinates of the beacons. Each coordinate is specified by two integer numbers d_i and l_i $(-90 \le d_i \le 90, -180 \le l_i \le 180)$, where d_i is the latitude, l_i is the longitude, both measured in degrees. The north latitude is positive, and the south latitude is negative. The west longitude is positive, and the east longitude is negative. The next line contains the Cartesian coordinates (x_1, y_1, z_1) of the Avenger starting point in the circummartian space The next line contains the direction (v_x, v_y, v_z) of the Avenger's movement.

All coordinates do not exceed 10 000 by absolute value. The North Pole is towards $(0, 0, 1)$ direction from the Martian center. The Prime Meridian crosses the Equator at $(1, 0, 0)$ direction from the Martian center. The beacon landing points do not belong to one great circle. Also there exists a point in space, visible from all beacon landing points.

Since the spacecraft moves really fast, you may disregard the Martian rotation and orbital movement and think of it as a fixed sphere. Gravitational attraction of Mars is also considered negligible.

Output

If Avenger successfully crashes inside or on the boundary of the safe landing area, output "YES", otherwise output "NO". It is guaranteed that if Avenger crashes outside of the safe landing area the distance between the crash point and the border of the safe landing area is at least $R \cdot 10^{-6}$.

Example

stdin	stdout
0 0 0 1 0 0 90 0 0 90 1 1 1 -1 -1 -1	YES

Problem G. Genesis Project

Time limit: 1 second
Memory limit: 128 MiB

A new type of bacteria is discovered by researchers of the top secret laboratory of Plasma-Synthesized Micro Formations. They are called "Multiproductors". Dr. Bacterius, the leading researcher, gives an interview and reveals interesting facts about Multiproductors:

- Multiproductors live in the two-dimensional space.
- Multiproductors can't move.

  And the most interesting...
- The Multiproductor's life span is only one day, after which it instantly dies. But right before this moment any two distinct multiproductors give birth to one more multiproductor in the middle of the segment connecting them.

"Oh, man!! They are like rabbits!" shouted out Dr. Bacterius.

"Yeah... But it is possible that two multiproductors are born at the same place! What happens then?" Dr. Virus asked.

Help researchers finding out the minimum number of days after which two multiproductors are born at the same point.

Input

The first line of the input contains the number of bacteria N $(1 \le N \le 1000)$. Each of the next N lines contains two integer numbers x_i, y_i — the coordinates of the i-th bacterium $(-10^9 \le x_i, y_i \le 10^9)$. Bacteria are so small that can be considered as points. Initially no two bacteria are located at the same point.

Output

Output 0 if two bacteria will never be born at the same point. Otherwise output the minimum number of days after which this happens.

Example

stdin	stdout
3 0 0 1 0 0 1	0
4 0 0 1 0 0 1 1 1	1
4 1 1 5 3 7 4 9 5	2

Problem H. Hadron Trip

Time limit: 3 seconds
Memory limit: 128 MiB

After a number of experiments at the Large Hadron Collider a group of scientists has developed an Autonomous Connectivity Module (ACM) that allows traveling at an incredible speed.

The first prototype instance of the ACM is ready, and the scientists want to use it to improve the quality of the global communication network. They consider N major cities (numbered from 1 to N). Some of these cities are connected by airline flights. Each flight connects a pair of cities and allows traveling between them (in both directions) in a given time.

The travel time between cities A and B is the minimum time to travel between A and B, possibly having connections at some intermediate cities. If city B can't be reached from city A at all, the travel time is assumed to be infinite. The maximum travel time among all pairs of the cities is an important factor, which is referred to as BFG (Breadth Factor of the Globe). Clearly, BFG can be infinite.

The scientists want to use ACM to establish a regular high-speed connection between a single pair of the cities to reduce BFG as much as possible.

When using ACM, the travel time depends only on the energy level of the device. ACM uses its own energy supply unit called ICPC (Instant Connection Power Consolidator). The bigger the ICPC energy level, the smaller the travel time. The latter may even be reduced to zero! Also, this time may be infinte.

Your goal is to help the scientists determining the minimum value of BFG that can be achieved by adding an ACM connection between one pair of cities. It is possible to add an ACM connection between two cities that are alredy connected via airline flight. Also, to save energy, you need to find the maximum time to travel through ACM that still allows achieving the minimum BFG.

Input

The first line of the input contains two numbers: N $(1 \le N \le 75)$, which is the number of cities and M $(0 \le M \le 1\,000)$ — the number of flights. Each of the following M lines corresponds to a single flight and contains three integers: two city numbers connected by this flight and the flight duration t $(0 \le t \le 10^8)$. No flight connects a city to itself, but several flights may connect the same pair of cities.

Output

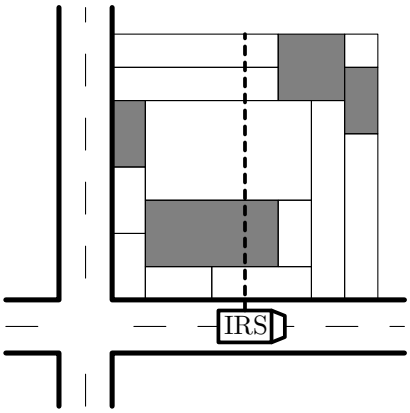
Output the minimum BFG that can be achieved and the maximum travel time of the ACM connection that allows attaining the minimum BFG. Both numbers must be accurate up to 0.00001. Write "`-1.00000`" (without quotes) instead of infinite values.

Example

stdin	stdout
5 5 1 2 1 3 2 1 2 4 2 2 5 2 4 5 4	3.00000 3.00000
4 1 1 2 0	-1.00000 -1.00000
4 2 1 2 1 4 3 1	2.00000 0.00000

Problem I. Mr. Incognito's Real Estate

Time limit: 1 second
Memory limit: 128 MiB



The construction company develops a new elite cottage settlement near the crossing of Begging Drive and Prison Road. The construction site has a rectangular shape with the borders directed along the roads. The site is divided into several rectangular plots of different sizes. Each piece of settlement land belongs to one and only one plot. The borders of the plots are parallel to the roads.

Due to the financial crisis the only man able to purchase real estate in the settlement is oligarch I (Incognito), as the crisis has not struck his business yet. He has already purchased two plots and wants to buy more.

After this additional purchase all Mr. I's plots should be *looking good*, which means that Mr. I's guests driving along atleast one of the roads and looking from their car perpendicularly to the road should see exactly one of Mr. I's plots during all the way along the border of the settlement (except for the moments when the line of view crosses the boundaries of two plots).

Help Mr. I to choose the plots he should purchase or report that it is impossible.

Input

The input first line contains two integer numbers X and Y — the width and the height of the settlement map $(0 < X, Y \le 10000)$. The second line contains an integer number N — the number of settlement plots $(0 < N \le 1000)$. Each of the following N lines describes single plot by four integer numbers x_1, y_1, x_2, y_2 — the coordinates of its bottom left and top right corners $(0 \le x_1 < x_2 \le X, 0 \le y_1 < y_2 \le Y)$. The origin of the map is located at the intersection of the roads. The plots are numbered from 1 to N in order of their appearance in the input data. The last line of the input contains the numbers of two plots already owned by the oligarch.

Output

The first output line must contain the total number of plots owned by the oligarch after the purchase. The second line must contain the serial numbers of all Mr. I's plots after the purchase in the order of visibility while driving along one of two roads from the crossing of the roads. If several solutions exist, output any of them.

If such a purchase is impossible output a single number -1.

Example

stdin	stdout
4 4 7 2 0 4 1 3 1 4 3 1 1 3 3 1 3 4 4 1 0 2 1 0 0 1 2 0 2 1 4 2 5	3 5 2 4

Problem J. Jig-saw Puzzle

Time limit: 4 seconds
Memory limit: 128 MiB

It's of no secret that famous illusionist Bavid Dlaine had become well-known because of his unique match magic. Few people, though, remember that he started his career with a very simple hocus-pocus described as follows. He turned away and asked a spectator to lay a few matches depicting a valid numerical equality with Roman numbers on a table. Then another spectator is asked to move exactly one match. After that he turned back and promptly pointed to this moved match.

You have decided to reveal the secret of Bavid and prove that there is no magic in his tricks. All you need is just a small program that solves such match problems.

Roman numbers are written as follows. There are four digits: `I` (letter i), `V`, `X`, and `L`. `I` denotes one, `V` denotes five, `X` denotes ten, and `L` denotes fifty. A natural number is represented by a sequence of digits as follows. No digit can occur more than three times in a row. A number larger than ten are written by giving its tens first, followed by the remaining units. If a larger digit comes before a smaller one they are summed up (summation principle); and if a smaller one comes before a larger then the smaller digit is subtracted from the larger (subtraction principle). The sequence of digits does not contain digits A and B preceding some third digit C, such as $A < C$ and $B < C$.

Bavid did not use large numbers in his trick, so the character `L` cannot occur more than once.

For example, the number `4` is written as Roman `IV`, but not `IIII`. The number `30` is written as Roman `XXX`, but not `XXL`. The number `45` is written as Roman `XLV`, but not `VL`. Finally, the number `100` is too large for Bavid, as it cannot be represented according to the rules specified above. Obviously, any number has no more than one Roman representation.

An *equality* is defined by the following grammar:

```
<equality>  ::= <statement> = <statement>
<statement> ::= <number>
<statement> ::= <number> + <statement>
<statement> ::= <number> - <statement>
```

Where `<number>` is a valid Roman number according to the rules above.

An equality is called *valid* if the number in its left-hand side equals to the number in its right-hand side.

Seven characters are used in equalities: `I`, `V`, `X`, `L`, `+`, `-`, `=`. These characters are constructed from matches as follows: for the characters `I` and `-` one match is needed, for the characters `V`, `X`, `L`, `+`, and `=` two matches are needed.

In the Bavid trick people were asked to move exactly one match. For a match to be moved it must be first taken from its character and then laid to a new position. A match is either laid over an existing character (including the one from which the match was taken), or laid between two consequent characters, or before the first character, or after the last one thus forming a new character. If a match is taken from a one-match character, this character vanishes and its left and right neighbors become adjacent. If a match is taken from a two-match character, the remaining matches must still represent some valid character. Here is the list of all possible extractions:

- `L` → `I` (the horizontal match is taken)
- `+` → `I` (the horizontal match is taken)
- `+` → `-` (the vertical match is taken)
- `=` → `-` (any horizontal match is taken)

There are two ways to lay a match between two consequent characters (and also before the first and after the last characters): one may either put I or -. Insertion of a match into a character is permitted if the character into which the match is being inserted remains valid. Here are all possible insertions of this kind:

- I → L (the horizontal match is laid)
- I → + (the horizontal match is laid)
- - → = (the horizontal match is laid)
- - → + (the vertical match is laid)

Also, it is possible to lay a match over the character from which it has been taken:

- I ↔ -
- L ↔ +
- = ↔ +
- V ↔ X

Write a program that finds all valid equalities that can be obtained from the input string by moving exactly one match. It is guaranteed that at least one valid equality exists.

Input

The first and the only line of input contains a string consisting of the characters I, V, X, L, +, -, and =. The length of the string does not exceed 100. It is guaranteed that the input string is not a valid equality as defined above.

Output

Output all different valid equalities that can be obtained from the input string by moving exactly one match. It is guaranteed that at least one valid equality exists.

Example

stdin	stdout
I=I-II	I=II-I
III=-II	III=III I+I=II
V-XX=I-XI	V-XV=I-XI X-XX=I-XI
II-I=I=I	II-I=II-I II-II=I-I III-I-I=I

Problem K. Klondike Gluon Rush

Time limit: 1 second
Memory limit: 128 MiB

The Gold Rush times are back!

After a series of highly successful experiments at the Large Hadron Collider a brand new technology of boson metallurgy has emerged. Boson metallurgy needs gluon ore as raw material. Gluon ore is extremely rare on Earth but, fortunately, very rich deposits of gluon ore are found on the bank of the Klondike River.

The bank of the Klondike River is divided into N lots forming an $N \times 1$ strip of lots $l_1, \ldots, l_N$.

All lots are initially claimed by M individuals. But there can be only one! So the owners decide to organize a poker tournament. In each day of the tournament a game of Higgs Poker is played. The winner chooses a segment of lots $[l_i, l_j]$ completely belonging to one of the losers and takes ownership on this segment. The tournament ends when the whole river bank is seized by one individual.

Calculate the minimum possible duration of the Higgs Poker Tournament.

Input

The first line of the input contains integer numbers N and M: the number of lots and the number of individuals respectively $(1 \le M \le N \le 300)$. The next line contains N numbers designating the original owners of the corresponding lots. Owners are numbered from 1 to M. Initially each individual owns at least one lot.

Output

Output the minimum possible duration of the Higgs Poker Tournament, in days.

Example

stdin	stdout
5 3 3 2 1 1 3	2

This answer may be obtained using the following scenario. In the first day the individual 2 wins the poker game and claims the lots 3–4. In the second day the individual 3 wins the poker game and claims lots 2–4. Other scenarios are also possible.

Problem L. Lucky Bonds

Time limit: 1 second
Memory limit: 128 MiB

Federal Treasury Bonds once were one of the best possible investment, but because of crisis their reputation is seriously staggered. So the Federal Treasury has developed a plan to regain the confidence of investors. The Treasury is willing to buy back series of unlucky bonds with consecutive numbers. Unfortunately, due to a computer error, only bonds with no leading zeroes in numbers are accepted.

A bond number consists of $2 \cdot N$ decimal digits. A bond is called *lucky* if the sum of the first N digits equals to the sum of the last N digits, and *unlucky* otherwise.

Given N, calculate the numbers of the first and the last bonds forming the longest consecutive series of unlucky bonds.

Input

Input consists of one integer number N $(1 \le N \le 10)$.

Output

Output the numbers of the first and the last bonds, one number per line forming the longest consecutive series of unlucky bonds. If several solutions exist, output any of them.

Example

stdin	stdout
1	67 76