2018 ECNA Regional Contest

ECNA 2018

November 10

icpc International Collegiate Programming Contest

north america sponsor

programming tools sponsor

2018 ICPC East Central North America Regional Contest

Problems

A	Car Vet
B	Difference
C	Pear-wise Voting
D	Pizza Cutting
E	The Punctilious Cruciverbalist
F	A Random Problem
G	Roman Holidays
H	Sequential Yahtzee
I	Tours de Sales Force
J	Watch Where You Step

Do not open before the contest has started.

International Collegiate Programming Contest
2018 East Central Regional Contest
Grand Valley State University
University of Cincinnati
University of Windsor
Youngstown State University
November 10, 2018

Rules:

1. There are **ten** problems to be completed in **5 hours**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. When displaying results, follow the format in the Sample Output for each problem. Unless otherwise stated, all whitespace in the Sample Output consists of exactly one blank character.
4. The allowed programming languages are C, C++, Java, Python 2 and Python 3.
5. All programs will be re-compiled prior to testing with the judges' data.
6. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
7. Programs will be run against multiple input files, each file containing a single test case.
8. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
9. All communication with the judges will be handled by the Kattis environment.
10. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A
Car Vet

Bob Roberts is the self proclaimed "Car Vet" who owns several junk car lots where people can come and search for spare parts for their (ailing) cars. The cars in each lot are parked in a fenced-in $m \times n$ grid lot, where each car takes up two grid squares. Each lot also has zero or more grid locations devoted to piles of parts (fenders, 8-track tape players, wheel covers, fuzzy dice, etc.); these locations are effectively blocked. Business has been so good that each of Bob's lots has only one empty grid space left.

From time to time, Bob or one of his lot attendants drops a part on the ground and it rolls underneath one of the cars. The only way to retrieve it is to move the car out of the way. Depending on the location of the empty grid space, it may be necessary to move several cars in order to achieve this. Cars can only move forward or backward, and car movement is also constrained by the fence, which prevents any car from moving off the lot, and by the blocked grid locations containing piles of parts.

Figure A.1 shows an example. An ambihelical hexnut has rolled under car number 3 in row 3, column 3 of the grid (shaded dark gray). The space in row 1, column 3 is empty and the space in row 3, column 4 is blocked. The only way to retrieve the part is to move car 8, then car 4, then car 3. Note that if the locations of the empty grid cell and the blocked grid cell were reversed, it would not be possible to retrieve the part.

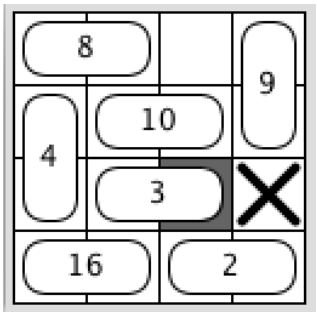

Figure A.1: Sample lot. Cars 8, 4, and 3 must be moved, in that order, to uncover the grid location in row 3, column 3. This corresponds to Sample Input 1.

The problem here should be obvious: For a given location Bob would like to know how to move cars around to uncover that location, or if it is even possible to uncover it.

Input

Input starts with a line containing two positive integers m n $(m, n \leq 250)$ indicating number of rows and columns of the junk car lot. Following this are m lines each containing n integers; the j^{th} value on the i^{th} line indicates the contents of the grid square at row i, column j. All values v are in the range $-2 \leq v \leq 62\,500$. Each non-negative value indicates that half of a junk car is in that location. Every non-negative value appears exactly twice and these two occurrences are horizontally or vertically adjacent to each other. A value of -1 indicates the empty grid location and a value of -2 indicates a blocked location. There is always exactly one empty location and at least one car, but there may be zero or more blocked locations. Following these m lines is a single line containing two integers r c $(1 \leq r, c \leq 250)$ indicating the row and column number of the desired location to be uncovered. This will always correspond to the location of a junk car.

Output

Display, on a single line separated by spaces, the numbers of the cars that must be moved to make the given location empty. They should appear in the order in which they must be moved. If there is more than one sequence of possible moves, display the sequence of shortest length. If there is still a tie, display the sequence that comes lexicographically first according to the car numbers in the sequence. If it is not possible to uncover the desired location, display `impossible`.

Sample Input 1

```
4 4
8 8 -1 9
4 10 10 9
4 3 3 -2
16 16 2 2
3 3
```

Sample Output 1

```
8 4 3
```

Sample Input 2

```
4 4
8 8 -2 9
4 10 10 9
4 3 3 -1
16 16 2 2
3 3
```

Sample Output 2

```
impossible
```

Problem B
Difference

A *smallest different sequence* (SDS) is a sequence of positive integers created as follows: $A_1 = r \geq 1$. For $n > 1$, $A_n = A_{n-1} + d$, where d is the smallest positive integer not yet appearing as a value in the sequence or as a difference between two values already in the sequence. For example, if $A_1 = 1$, then since 2 is the smallest number not in our sequence so far, $A_2 = A_1 + 2 = 3$. Likewise $A_3 = 7$, since $1, 2$ and 3 are already accounted for, either as values in the sequence, or as a difference between two values. Continuing, we have $1, 2, 3, 4, 6$, and 7 accounted for, leaving 5 as our next smallest difference; thus $A_4 = 12$. The next few values in this SDS are $20, 30, 44, 59, 75, 96, \ldots$ For a positive integer m, you are to determine where in the SDS m first appears, either as a value in the SDS or as a difference between two values in the SDS. In the above SDS, $12, 5, 9$ and 11 first appear in step 4.

Input

Input consists of a single line containing two positive integers A_1 m ($1 \leq r \leq 100, 1 \leq m \leq 200\,000\,000$).

Output

Display the smallest value n such that the sequence $A_1, \ldots, A_n$ either contains m as a value in the sequence or as a difference between two values in the sequence. All answers will be $\leq 10\,000$.

Sample Input 1	Sample Output 1
1 5	4

Sample Input 2	Sample Output 2
1 12	4

Sample Input 3	Sample Output 3
5 1	2

This page is intentionally left blank.

Problem C
Pear-wise Voting

Bob Roberts is ending his term as president of a local group of fruit aficionados call the Pear-wise Club. Being president is a plum position so many people are running, some of whom Bob thinks are just peachy, while others are just the pits. After some thought, Bob has amended the by-laws of the Pear-wise Club to use sequential pairwise voting to determine his successor.

Sequential pairwise voting works as follows: every voter presents a ballot which ranks the candidates from most to least favorable. Then two of the candidates are selected to hold a one-on-one contest with these ballots, using the relative positions of the candidates on each ballot to determine how each voter would vote in this contest. The winner of this contest then takes part in a second one-on-one contest with a third candidate; the winner of that takes part in a third contest with a fourth candidate, and so on. The winner of the election is the winner of the last one-on-one contest.

The order in which the candidates take part in the one-on-one contests is specified by the *agenda*, which is a pre-selected ordering of the candidates. The first two candidates on the agenda take part in the first contest; the winner of that contest then faces the third candidate on the agenda, an so on. For example, assume we had the following set of 13 ballots, each ranking five candidates A, B, C, D and E:

(4)	(3)	(3)	(2)	(1)
A	D	C	B	E
D	C	A	D	B
C	A	D	C	C
B	B	B	A	D
E	E	E	E	A

The number above each ballot indicates the number of voters who submitted that ballot. If the agenda is ABCDE, then sequential pairwise voting would proceed as follows: in the first contest between A and B, A wins 10–3; A would then face C (the third candidate on the agenda) and here C wins 9–4; C would then face D and D wins 9–4; and finally D would face E and D would win 12–1, so D would be the overall winner using this agenda. Using the reverse agenda – EDCBA – would result in A winning the election (we'll let you figure out the details of that one).

Now it wasn't out of any political science reasons that Bob picked sequential pairwise voting, or the coincidence of its name. He knows that the results are highly dependent on the agenda that's used and as current president, Bob gets to set the agenda! Since he's also aware of everyone's voting preferences, he's pretty sure that he can find an agenda so that his preference for who succeeds him will be elected. Just to be sure, he would like you to write a program which, when given a set of ballots as input, determines, for each candidate, whether there is an agenda which will result in his/her winning the election.

Input

Input begins with a line containing two positive integers n m ($n \leq 26, m \leq 2\,000$) indicating the number of candidates and the number of unique ballots, respectively. Candidates are labeled with the first n letters of the alphabet, uppercase. Each of the next m lines displays one of the m ballots. Each line starts with a

positive integer p ($p \leq 100$) indicating the number of voters who submitted the ballot, following by a string of n characters specifying the ballot. This upper-case string is a permutation of the first n characters of the English alphabet, with the first character in the string indicating the most favored candidate in the ballot, the second character the second-most favored, and so on. The total number of votes over all ballots will be odd, so there can't be a tie in any pairwise vote.

Output

For each candidate display the candidate's letter followed by a colon, followed by either the phrase `can win` if an agenda exists where the candidate can win, or the phrase `can't win` if no such agenda exists.

Sample Input 1

```
5 5
4 ADCBE
3 DCABE
3 CADBE
2 BDCAE
1 EBCDA
```

Sample Output 1

```
A: can win
B: can't win
C: can win
D: can win
E: can't win
```

Sample Input 2

```
3 3
1 ABC
1 BCA
1 CAB
```

Sample Output 2

```
A: can win
B: can win
C: can win
```

Problem D
Pizza Cutting

Pizzas are usually cooked as a circle, and are often cut "triangularly" along lines moving out from the center, as shown on the left in Figure D.1 below. However, some heretical people prefer that the pizza be cut into rectangular pieces, as shown on the right in the figure:

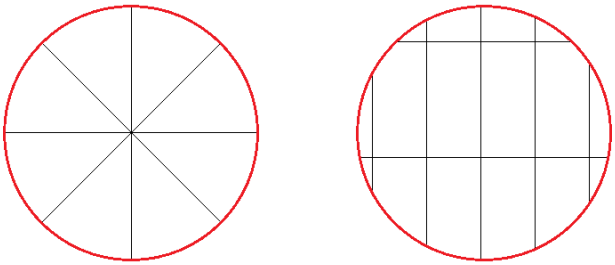

Figure D.1: Sample pizza cuts. The pizza on the right corresponds to Sample Inputs 1 and 2.

As you can see, besides the obvious aesthetic issues involved, the pieces that intersect the boundary of the circle may not be complete rectangles. Sometimes these pieces are close to the largest size piece, but other times they are so small as to be useless. Given a pizza radius, lengths and widths of the rectangular cuts, and a "too small" percentage (for example, any slice that is less than 50% of the largest slice is too small), how many slices are cut that will be "too small"?

Input

Input consists of a single line containing six numbers r dx dy x y p, where r $(1 \leq r \leq 1\,000)$ is the radius of the pizza, dx and dy $(1 \leq dx, dy \leq 3\,000)$ are the width and height of the pizza cuts, x and y $(-10\,000 \leq x, y \leq 10\,000)$ is the intersection point of one vertical and one horizontal cut, and p $(0.0 < p < 1.0)$ is the percentage defining a "too small" piece. All values are integers except for p, which is a real number with at most 3 decimal places. The pizza is centered at point $(0, 0)$.

Output

Display the number of pieces of pizza that are less than p times the size of the largest piece. A piece with area X is considered p times smaller than another piece with area Y if |X/Y - p |$\leq 10^{-6}$.

Sample Input 1	Sample Output 1
100 45 90 0 -20 0.1	4

Sample Input 2	Sample Output 2
100 45 90 0 -20 .999	14

This page is intentionally left blank.

Problem E
The Punctilious Cruciverbalist

A crossword puzzle consists of a grid of squares in which intersecting words are placed, one letter per square. Some grid squares are filled black indicating that no letter should go there. Solvers are given clues for each word, and each clue is identified by a starting clue number and either the word Across (for horizontal words) or Down (for vertical words). Each square that can start a word is given a clue number, starting in the upper left and proceeding left-to-right, row-wise. A square will contain an across-clue number if the square to its left is either black or off the grid, and a square will contain a down-clue number if the square above it is either black or off the grid. See Figure E.1 for a simple example.

Will Longz is an avid crossword puzzle enthusiast and as with most solvers, he first tries to solve clues where some of the letters are already present (from already solved clues whose answers intersect the clue he is working on). Typically, the closer to the beginning of the word that these letters are, the better. Will has developed a metric to help him decide the order in which he should solve clues. If the answer to a clue spans n squares, he assigns the value n to the first of these squares, $n-1$ to the second square, and so on ending with assigning 1 to last square. The rank of any unsolved clue is then defined as follows: for each square that already has a letter in it, add the value assigned to that square to a running total. Then divide this total by the sum of all the values assigned to that clue to get that clue's rank. Once this is done for all the clues, Will solves the clue with the highest rank. If there is a tie, he selects an across clue over a down clue. If there is still a tie he picks the clue with the smallest starting clue number. After each clue is solved Will recalculates the rankings before selecting the next highest ranked clue.

As an example, consider the crossword in Figure E.1 where the 6-Across clue has already been solved (we'll used 6A as a shorter specification for this clue). To determine the rank of the 1-Down (1D) clue, Will first assigns the values 3, 2 and 1 to each of the squares going down. Since the last square has a letter in it, the ranking of this clue is $1/(3+2+1) = 1/6$. In a similar fashion he determines that the ranking of clues 2D and 3D are $2/10$ and the ranking of 5D is $2/6$. The rankings of the three unsolved across clues 1A, 4A and 7A are all 0. Thus the next clue to solve is 5D (we'll be optimistic and always assume Will can solve any clue). Once this is solved, the rankings of 4A and 7A become $1/10$ and $1/6$, respectively. Since there is now a tie between the highest ranking clues (2D and 3D) and they are both down clues, Will picks the down clue with the lowest clue number and solves 2D. Proceeding similarly, the remaining clues are solved in the following order (with their ranking at time of solution in parentheses): 7A $(4/6)$, 4A $(4/10)$, 3D $(6/10)$ and 1A $(3/6)$. Note that clue 5D is not on the list, as it is completely solved by the previously solved clues (namely 4A, 6A and 7A).

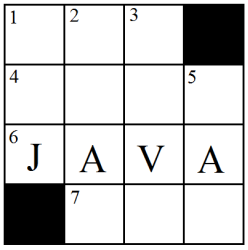

Figure E.1: Example crossword puzzle.

For this problem, you will be given a crossword puzzle grid with zero or more squares already filled in and you must determine the order in which unsolved clues should be solved. The filled-in squares may or may not correspond to completely solved clues. One final twist: we will only give you the black squares and letters in the puzzle. You must determine the clue numbers (I bet Will could do it!).

Input

Input starts with a line containing two integers r c ($1 \leq r, c \leq 50$) specifying the number of rows and columns in the crossword grid. Following this are r lines each containing c characters which describe the puzzle. Each of these characters is either a '`.`' (indicating an empty square), a '`#`' (indicating a black square), or an uppercase alphabetic English character (indicating a solved square). There will always be at least one empty square in the grid.

Output

Display the order in which clues should be solved one per line, using the metric described above. Each clue should start with the clue number followed by either the letter '`A`' or '`D`' (for an across or down clue).

Sample Input 1	Sample Output 1
4 4 ...# .... JAVA #...	5D 2D 7A 4A 3D 1A

Problem F
A Random Problem

Generating a random number sequence is not easy. Many sequences may look random but upon closer inspection we can find hidden regularities in the occurrence of the numbers. For example, consider the following 100-digit "random" sequence starting with `4 7 9...` :

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

If you look closely, whenever a 4 is followed immediately by another 4, the third value after this will be a 3 (we're sure you saw that). We'll call features like this *triple correlations* and we'll represent it as $4(1)4(3)3$, where the numbers in parentheses indicate how far away each number on either side of the parentheses must be. More precisely, a sequence of p digits has an $a(n)b(m)c$ triple correlation if

1. any time there is an a followed n locations later by a b there is always a c located m locations after the b unless b's location is within distance $m-1$ of the end of the sequence.
2. any time there is a b followed m locations later by a c there is always an a located n locations before the b unless b's location is within distance $n-1$ of the beginning of the sequence.
3. any time there is an a followed $n+m$ locations later by a c there is always a b located n locations after the a.
4. the correlation occurs at least $\lceil p/40 \rceil + 1$ times, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

Such correlations are tough to spot in general, so that's where we need your help. You will be given a sequence of digits and must search it for triple correlations.

Input

Input starts with a positive integer p ($p \leq 1\,000$) indicating the number of random digits in the sequence. Starting on the next line will be the p digits, separated by spaces and potentially spread across multiple lines. No line will contain more than 100 digits and there will be no blank lines.

Output

Display `triple correlation a(n)b(m)c found` if it occurs in the list (with the appropriate values for `a`, `b`, `c`, `n`, and `m` filled in) or the phrase `random sequence` otherwise. If there are multiple triple correlations in the sequence, display the one which has its first occurrence (as determined by the first

occurrence of its a-value) earliest in the sequence. If there is still a tie, choose the one with the smaller value of n, and if there is still a tie, choose the one with the smaller value of m.

Sample Input 1

```
100
4 7 9 5 9 3 5 0 0 1 7 8 5 0 2 6 3 5 4 4
4 6 3 3 2 7 1 8 7 8 7 6 1 1 7 2 5 4 7 2
0 4 4 5 8 3 0 6 9 3 2 6 6 8 5 2 5 1 2 7
2 4 1 0 0 4 9 1 8 7 5 0 4 4 8 4 3 2 6 8
8 5 6 7 0 9 7 0 3 6 1 4 4 1 2 3 2 6 9 9
```

Sample Output 1

```
triple correlation 4(1)4(3)3 found
```

Sample Input 2

```
10
1 2 3 1 2 2 1 1 3 0
```

Sample Output 2

```
random sequence
```

Problem G
Roman Holidays

The ancient Romans created many important things: aqueducts, really straight roads, togas, those candles that spout fireworks. But the most useless is Roman numerals, a very awkward way to represent positive integers.

The Roman numeral system uses seven different letters, each representing a different numerical value: the letter I represents the value 1, V 5, X 10, L 50, C 100, D 500 and M 1 000. These can be combined to form the following *base* values:

1	2	3	4	5	6	7	8	9	10
I	II	III	IV	V	VI	VII	VIII	IX	X

10	20	30	40	50	60	70	80	90	100
X	XX	XXX	XL	L	LX	LXX	LXXX	XC	C

100	200	300	400	500	600	700	800	900	1 000
C	CC	CCC	CD	D	DC	DCC	DCCC	CM	M

The Roman numeral representation of a non-base value number x is obtained by first breaking up x into a sum of base values and then translating each base value, largest to smallest. When choosing base values you always choose the largest one $\leq x$ first, then the largest one $\leq$ the amount remaining, and so on. Thus $14 = 10 + 4$ = XIV, $792 = 700 + 90 + 2$ = DCCXCII. Numbers larger than 1 000 use as many M's as necessary. So 2 018 = MMXVIII and 1 000 000 would be a string of one thousand M's (hence the word "awkward" in the first paragraph).

The Roman numeral representation gives a new way to order the positive integers. We can now order them alphabetically if we treat the Roman representation of each integer as a word. If one word A is a prefix for another word B then A comes first. We'll call this the *roman ordering* of the positive integers. Thus the first number in roman ordering is `C` (100 in our system). The next three numbers would be `CC`, `CCC` and `CCCI`, and so on.

Note in roman ordering, all numbers larger than 1 000 would come before any number starting with `V` or `X`. Indeed the last number is `XXXVIII`. In this problem you will be given one or more positive integers and must determine their positions in the roman ordering – from the front or back as appropriate.

Input

Input starts with a positive integer $n \leq 100$ indicating the number of positive integers to follow, each on a separate line. Each of these remaining numbers will be $\leq 10^9$.

Output

For each value (other than n), output the position of the integer in the roman ordering, one per line. If the position is relative to the end of the roman ordering, make the integer negative. Thus 38 has roman ordering position -1, 37 has position -2, and so on.

Sample Input 1	Sample Output 1
3 100 101 38	1 302 -1

Problem H
Sequential Yahtzee

Danny has a hand-held game of Yahtzee that he likes to play (how 90's!). The object of Yahztee is to score points by placing the result of rolls of 5 dice into one of 13 categories. The categories are listed below, along with how you score points for each category:

Category	Scoring	Category	Scoring
1's	1 point for each 1	3-of-a-Kind	total of all 5 dice
2's	2 points for each 2	4-of-a-Kind	total of all 5 dice
3's	3 points for each 3	Full House	25
4's	4 points for each 4	Small Straight	30
5's	5 points for each 5	Long Straight	40
6's	6 points for each 6	Chance	total of all 5 dice
		Yahtzee	50

A 3(or 4)-of-a-Kind is any 5 dice where at least three (or four) show the same value. A Full House consists of 3 dice with the same value and the other two with the same value (different from the first value); a Small Straight is four consecutive values on any of four of the dice, a Long Straight is five consecutive values, and a Yahtzee is all five dice showing the same value. Finally the Chance category can be used for any set of five dice. For example, if the five dice showed four 2's and one 5, you would score 8 points if you put it in the 2's category, 5 points if you put it in the 5's category, and 13 points if you put it in either the 3-of-Kind, 4-of-a-Kind or Chance categories. If you put it in any other category you would get 0 points.

A game consists of 13 rounds. At the start of each round, you roll all 5 dice. You can either assign the 5 dice to any of the unused categories or (more likely) re-roll any number of the dice (even all 5 if you like). You get to do this one more time and after (at most) the third roll you must place the dice in an unused category. After 13 rounds, all the categories have been used and you total up the points you got for each category.

In regular Yahtzee you have your choice of which of the scoring categories to use after every round – in fact, the decision on which category to use is one of the challenges of Yahtzee. Danny normally plays this way, but as we said he plays A LOT of Yahtzee, so sometimes he like to switch things up a bit. One of his favorite variations is something he calls *sequential yahtzee*. In this version, the only category you can use after the first set of rolls is the 1's (the first category on his hand-held game); after this, you must use the 2's for your second category, and so on (in the order the categories are given in the table above) until you reach the Yahtzee category.

For example, suppose there's a glitch in Danny's game and the dice only roll 1's (it is a pretty old game). After the first round Danny has (what else) a set of five 1's. In regular Yahtzee he could score 50 points for a Yahtzee, but in sequential yahtzee he must put it in the 1's category and scores 5 points. After he rolls five 1's again, he must put it in the 2's category and scores 0. He scores 0 for the next 4 rounds, putting his five 1's in the 3's, 4's, 5's and 6's category. He gets 5 points for each of the next two rounds, placing his five 1's first in the 3-of-a-Kind and then in the 4-Of-A-Kind. He gets nothing in the next two rounds, scores 5 points for the Chance category and then FINALLY gets 50 points for a Yahtzee in the 13th round. All together he scores 70 points

Danny keeps track of all the dice rolls in the game and often wonders if he could have done better than he

did in a game of sequential yahtzee. Your job is simple: given a sequence of consecutive dice rolls, what's the maximum score possible in a game of sequential yahtzee? Well, maybe it's just the description of the problem that's simple.

Input

Input starts with a line containing an integer n ($65 \leq n \leq 195$) indicating the number of dice rolls. Following this, on one or more lines, are the n dice rolls, all values between 1 and 6 inclusive.

Output

Display the maximum possible sequential yahtzee score using the given dice rolls, filling in all 13 categories. Not all the dice rolls need to be used, but those that are must be consecutive starting with the first roll.

Sample Input 1	Sample Output 1
65 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	70

Problem I
Tours de Sales Force

The Weight For It company sells home gym equipment to clients in multiple states. Each salesperson in a state is currently in charge of a specific district made up of 3-8 clients. Each salesperson has mapped out the shortest path with which to visit all of their clients in one day (this is known as their *sales tour*).

Sales have gotten a little light for Weight For It so they've decided to cut their sales force in half, reassigning each of the districts of the fired half of the workforce in one state to one in the un-fired half. In order to be fair, each of the un-fired sales persons are assigned exactly one of these districts. After the assignments the remaining salespeople will have to recalculate their minimum sales tours, and as a further cost cutting measure Weight For It would like to assign the districts so that the overall length of all these new sales tours is as small as possible.

For example, Figure I.1 on the left shows the configuration of four districts in one state. Suppose that the salespeople for districts A and B are fired. Then the best reassignment for those districts would be to assign the six clients from district A to the salesperson for district C and the four clients from district B to the salesperson for district D. The resulting shortest sales tours for each of the new districts C' and D' are shown on the right.

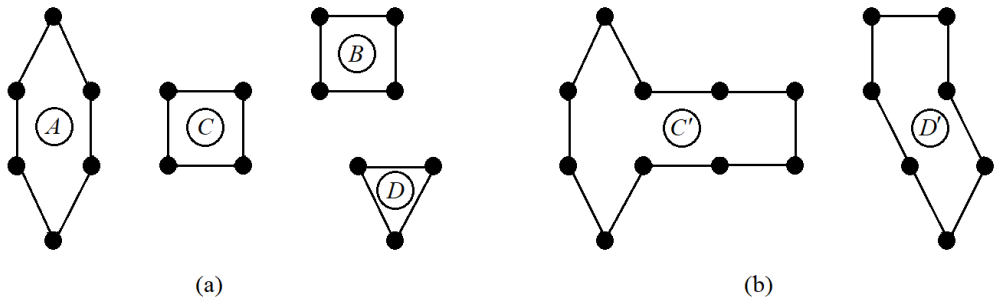

Figure I.1: (a) Four districts before firings. (b) Two districts after firings. This example corresponds to Sample Input 1.

Input

Input starts with a positive even integer d ($d \leq 50$) indicating the number of districts. Each of the next d lines lists the number and locations of the clients in each of the districts, one district per line starting with district one. Each of these lines starts with an integer n ($3 \leq n \leq 8$) indicating the number of clients in the district, followed by n integer coordinate pairs x y ($-10\,000 \leq x, y \leq 10\,000$) indicating the locations of the clients. The first $d/2$ districts listed are those of the fired salespersons. The distance between any two clients is the Euclidean distance and all client locations are distinct.

Output

Display the sum of all the sales tours prior to the firings followed by the sum of all the sales tours after the firings. Your answers should be correct to within an absolute error of 10^{-2}.

Sample Input 1	Sample Output 1
4 6 0 10 0 20 5 30 10 20 10 10 5 0 4 40 20 40 30 50 20 50 30 4 20 10 30 20 20 20 30 10 3 55 10 45 10 50 0	177.082039 179.442719

Problem J
Watch Where You Step

You are an employee at the local zoo and have recently been promoted from excretory extraction to overseeing the layout of all the sidewalks throughout the property. Currently all the sidewalks are 1-way, and the zoo is set up in various *zones*. Each zone is made up of a set of *attractions* (elephant enclosure, lizard house, etc.), and within any one zone you can get from any attraction to any other attraction in the zone using one or more sidewalks. Once you take a sidewalk from one zone to another you can never return to the zone you left. However, it is possible to walk to every zone in a single visit to the zoo. The original designers thought this arrangement was very important to help control the flow of visitors to the zoo.

The members of the board of directors have come to you with a problem. They agree with the original designers in the use of zones, but they feel that more one-way sidewalks could be included to make the zoo a little more patron-friendly. They would like you to figure out the maximum number of sidewalks that could be added so as not to introduce a path between any two attractions which didn't have one between them before.

For example, consider the small zoo shown in Figure J.1, with 7 attractions labeled 1 (the "Camel Castle") through 7 (the "Hippo Hippodrome"). Currently attractions 1, 2, 3 and 4 form one zone and 5, 6 and 7 form another. You can add sidewalks from 1, 2, 3 or 4 to either 5, 6 or 7, but adding a sidewalk from (say) 7 to 1 would allow patrons to get from 7 to 1, which was impossible previously. Note that you can also add sidewalks between all attractions within a zone which don't already exist (for example, from 1 to 3 or from 2 to 4). The total number of possible new sidewalks for this zoo would be 21.

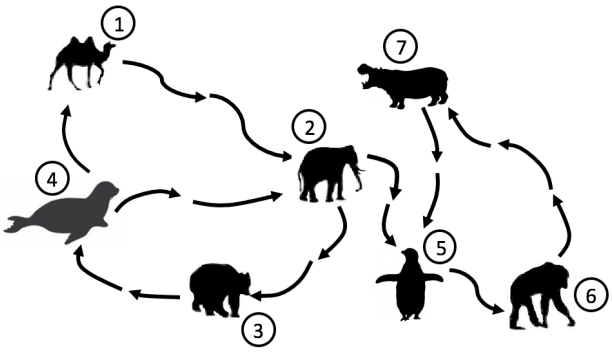

Figure J.1: Sample zoo. This example corresponds to Sample Input 1.

Input

Input starts with a line containing an integer n ($1 \leq n \leq 2\,500$), the number of attractions, labeled 1 to n. After this are n lines each containing n integers. If the j^{th} integer on the i^{th} of these lines is a 1 it indicates that there is a one-way sidewalk from attraction i to attraction j; otherwise this integer will be a 0 indicating no such sidewalk is present. There is never a sidewalk from an attraction to itself.

Output

Display the maximum number of new one-way sidewalks that could be added to the zoo.

Sample Input 1

```
7
0 1 0 0 0 0 0
0 0 1 0 1 0 0
1 0 0 1 0 0 0
1 0 0 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 1
0 0 0 0 1 0 0
```

Sample Output 1

```
21
```

Sample Input 2

```
5
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0
```

Sample Output 2

```
6
```

Sample Input 3

```
2
0 1
1 0
```

Sample Output 3

```
0
```