Indian National Olympiad in Informatics, 2002

Time: 3 hours *1 May, 2002*

Attempt all questions.

1. You are given n boxes, numbered $1, 2, \ldots, n$, and a number p, $1 \leq p \leq n$. You have to interchange the contents of boxes $p, p{+}1, \ldots, n$ with the contents of boxes $1, 2, \ldots, p{-}1$. More precisely, you are supposed to rearrange the contents of the boxes so that the original contents of boxes $p, p{+}1, \ldots, n$ are moved (in sequence) to boxes $1, 2, \ldots, (n{-}p){+}1$, and the original contents of boxes $1, 2, \ldots, p{-}1$ are moved (in sequence) to the boxes $(n{-}p) + 2, (n{-}p) + 3, \ldots, n$.

   Here are two examples.

   10 boxes, $p = 7$

	1	2	3	4	5	6	7	8	9	10
Before	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
After	a_7	a_8	a_9	a_{10}	a_1	a_2	a_3	a_4	a_5	a_6

   10 boxes, $p = 3$

	1	2	3	4	5	6	7	8	9	10
Before	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
After	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_1	a_2

   There is an important restriction on how you may rearrange the contents: at each step, the *only* operation available is to interchange (swap) the contents of any two boxes (for instance, you can choose box numbers 4 and 7 and move the contents of box 4 to box 7 and simultaneously move the contents of box 7 to box 4).

   Describe an algorithm to rearrange the contents of the boxes as required, using only this basic operation of swapping the contents of two boxes.

2. (a) Consider the following game involving two players, A and B. There is a stack of k coins. The players move alternately, with Player A moving first. In each move, a player can remove either 1 or 2 coins from the stack. The player who removes the last coin wins the game.

   Depending on the initial number of coins k, either Player A or Player B has a strategy that will guarantee a win. For what values of k will Player A win? For what values of k will Player B win? Describe the winning strategy in both cases.

(b) Consider a variation of the earlier game in which you have two stacks of coins that initially contain m and n coins, respectively. As before, the players move alternately, with Player A moving first. In each move, a player can remove either 1 or 2 coins from one of the two stacks (but if 2 coins are removed, both must be from the same stack). As before, the player who removes the last coin overall wins the game.

Once again, depending on the initial number of coins m and n, either Player A or Player B has a strategy that will guarantee a win. For what values of m and n will Player A win? For what values of m and n will Player B win? Describe the winning strategy in both cases.

3. (a) Our town is the terminus of a railway line. After passengers have disembarked, the carriages are rearranged and the train is prepared for departure.

The train yard has a peculiar arrangement, as shown in Figure 1. Trains arrive from the right. There is a *rearranging yard* that can rearrange 8 carriages at a time, in any order. However, the trains that run are normally 16 carriages long. After the rearranging yard, there are two long sidings that can hold up to 16 carriages each.

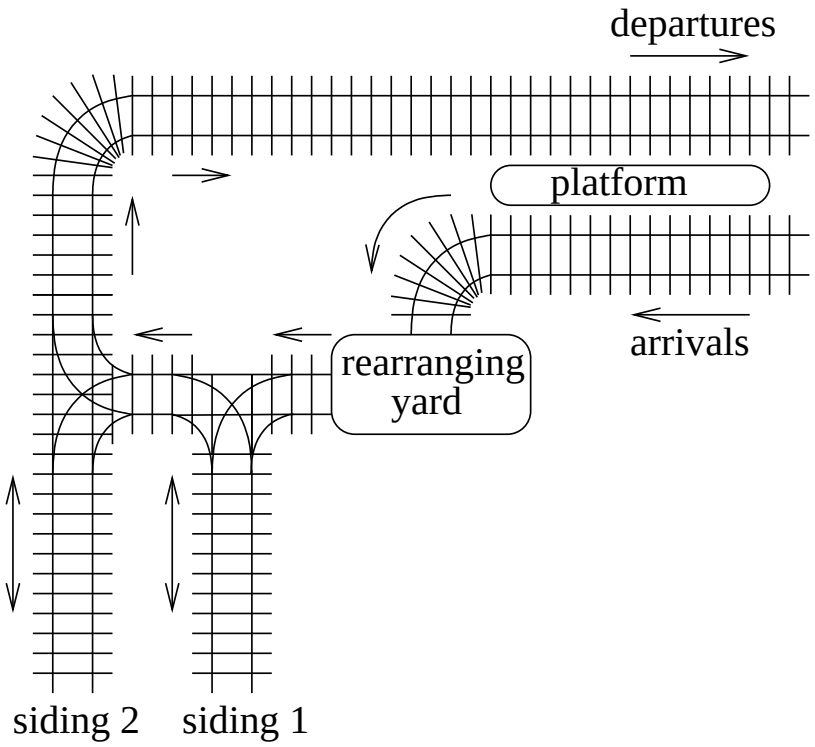

Figure 1:

Carriages can be added and removed one at a time from the two sidings, but carriages cannot move backwards from the second siding to the first, from the first siding back to the rearranging yard or from the rearranging yard back to the

platform. When the train arrives at the departure platform the carriages must be in the correct order.

Describe an algorithm for rearranging trains with 16 carriages in any order that one wants using this shunting arrangement. Note that the rearrangement may involve mixing up carriages from the first half of the train and the second half. For instance, the incoming train could be

$$c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10}c_{11}c_{12}c_{13}c_{14}c_{15}c_{16}$$

and the outgoing train could be required to be

$$c_1c_3c_5c_7c_9c_{11}c_{13}c_{15}c_2c_4c_6c_8c_{10}c_{12}c_{14}c_{16}.$$

(b) When upgrading the station, it was noticed that the rearranging yard could be dispensed with and replaced by additional sidings. Show that for any value of n, the arrangement of n sidings in Figure 2, where carriages can move from one siding to the next but not backwards, can be used to rearrange a train with 2^n carriages in any desired order. Assume that each siding is long enough to hold all 2^n carriages.

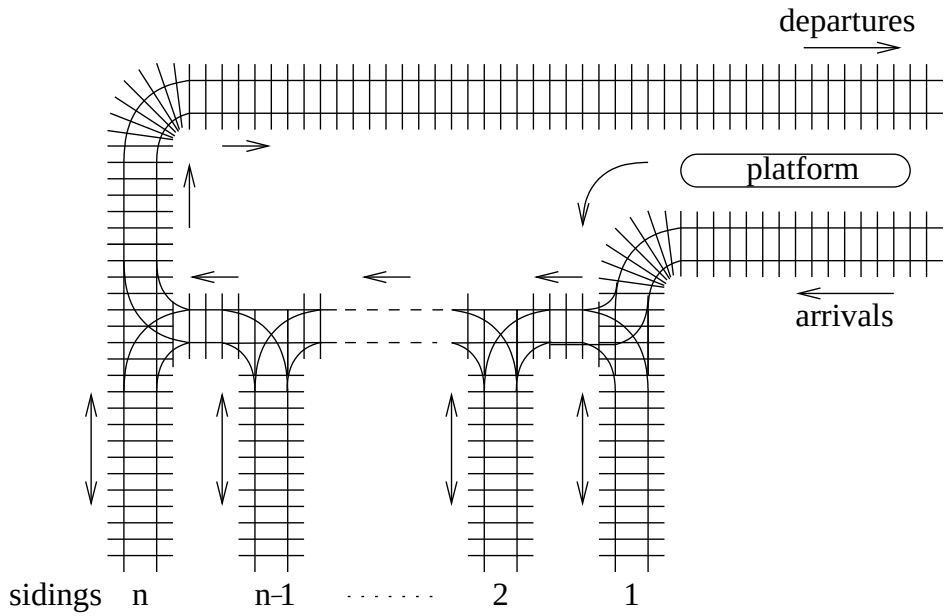

Figure 2:

Indian National Olympiad in Informatics, 2002
Suggested solutions to problems

1. There are at least two straightforward solutions.

   Non-recursive solution Let the original contents be (in sequence) $a_1 a_2 \ldots a_{p-1} a_p a_{p+1} \ldots a_n$. First, let a_p "bubble" forward to the first box as follows.

   - Swap contents of box p with box $p-1$. Contents are now $a_1 a_2 \ldots a_p a_{p-1} a_{p+1} \ldots a_n$.
   - Swap contents of box $p-1$ with box $p-2$. Contents are now $a_1 a_2 \ldots a_p a_{p-2} a_{p-1} a_{p+1} \ldots a_n$.
   - $\ldots$
   - Swap contents of box 2 with box 1. Contents are now $a_p a_1 a_2 \ldots a_{p-2} a_{p-1} a_{p+1} \ldots a_n$.

   Similarly, move a_{p+1} to box 2— repeat the earlier procedure starting with box $p+1$ and stopping with box 2 to get $a_p a_{p+1} a_1 a_2 \ldots a_{p-2} a_{p-1} a_{p+2} \ldots a_n$.

   Do the same for $p+2$ to 3, $p+3$ to 4, $\ldots$, n to $(n-p)+1$ so that eventually we have $a_p a_{p+1} \ldots a_n a_1 a_2 \ldots a_{p-1}$.

   Recursive solution

   Case 1 Let $p \geq n/2$. Swap the contents of box 1 with box p, box 2 with box $p+1$, $\ldots$, box $n-p+1$ with box n to get a new arrangement $a_p a_{p+1} \ldots a_n a_{n-p+2} \ldots a_{p-1} a_1 a_2 \ldots a_{n-p+1}$.

   Case 2 If $p < n/2$, instead swap the contents of box 1 with box $n-p+2$, box 2 with box $n-p+3$, $\ldots$, box $p-1$ with box n to get a new arrangement $a_{n-p+2} \ldots a_n a_{p+1} \ldots a_{n-p} a_1 a_2 \ldots a_{p-1}$.

   After the first step, in case 1 ($p \geq n/2$), recursively apply the same procedure for boxes $n-p+2, \ldots, n$ with appropriate new values $n' = n-p+1$ and $p' = n-2p+3$ so that contents $a_{n-p+2} \ldots a_{p-1}$ of the $n-2p+2$ boxes numbered $n-p+2, \ldots, p-1$ is interchanged with contents $a_1 a_2 \ldots a_{n-p+1}$ of the $p-1$ boxes numbered $n-p+1, \ldots, n$.

   Similarly, in the second case ($p < n/2$), recursively apply the procedure for boxes $1, \ldots, n-p$ with $n' = n-p+1$ and $p' = n-2p+3$ so that contents $a_{n-p+2} \ldots a_{n-2p}$ of the $n-2p+2$ boxes numbered $n-p+2, \ldots, p-1$ is interchanged with contents $a_1 a_2 \ldots a_{n-p+1}$ of the $p-1$ boxes numbered $n-p+1, \ldots, n$.

2. (a) If a player has to play when the stack has 3 coins, he loses—whether he takes 1 or 2 coins, the opponent can empty the stack on the next move and thus win the game.

   If a player plays when the stack has 6 coins, the opponent can leave the stack with 3 coins after his next move, leading to a losing position for the first player.

In general, it follows that any player who moves when the number of coins in the stack is a multiple of 3 loses— the opponent can reduce it each time to the next lower multiple of 3 until eventually the stack reaches exactly 3 coins, which is a losing position.

Conversely, if the number of coins is not a multiple of 3, then the player who moves can reduce it to a multiple of 3 and this is a losing position for the opponent.

Thus, if the number of coins in initial stack is *not* a multiple of 3, Player A has a winning strategy and if the number of coins *is* a multiple of 3, Player B has a winning strategy. The winning strategy in either case is to reduce the number of coins to the next lower multiple of 3.

(b) If there are two stacks, a losing position is one in which the *difference* between the two stacks is a multiple of 3. If a player plays when the difference is a multiple of 3, then the opponent can always restore the difference to be a multiple of 3. Eventually, when one of the two stacks becomes empty, the invariant guarantees that the opponent can make the number of coins in the nonempty stack a multiple of 3, which is a losing position for first player in the single stack game (by the first part of this question).

Therefore, Player A can always win if, in the initial arrangement, the difference between the number of coins in the two stacks is *not* a multiple of 3, and Player B can always win if the initial difference *is* a multiple of 3. The winning strategy in both cases is to restore the difference to a multiple of 3 so long as both stacks are nonempty and then switch to the winning strategy for a single stack after one of the stacks becomes empty. (Actually, the strategy does not really change after one stack becomes empty—when one of the stacks is empty, if the difference between the stacks is a multiple of 3 then the size of the nonempty stack must be a multiple of 3.)

3. (a) Move the first 8 carriages to the rearranging yard and rearrange them in the *reverse* order to which they would appear in the final arrangement. After rearranging them, transfer these 8 carriages into siding 1. Do the same for the next 8 carriages and move them into siding 2. Now, reassemble the final train into the correct order by *merging* the carriages from the two sidings, one at a time—that is, compare the next available carriage in both sidings and pull out the one that is to be placed next in the final arrangement.

For instance, in the example given in the question, the incoming train is arranged as

$$c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10}c_{11}c_{12}c_{13}c_{14}c_{15}c_{16}$$

and the outgoing train is required to be

$$c_1c_3c_5c_7c_9c_{11}c_{13}c_{15}c_2c_4c_6c_8c_{10}c_{12}c_{14}c_{16}.$$

The correct order for the first 8 carriages $c_1c_2c_3c_4c_5c_6c_7c_8$ in the final train is $c_1c_3c_5c_7c_2c_4c_6c_8$. We thus rearrange the carriages as $c_8c_6c_4c_2c_7c_5c_3c_1$ and push

them into siding 1, with c_1 as the outermost carriage. Likewise, we rearrange the next 8 carriages as $c_{16}c_{14}c_{12}c_{10}c_{15}c_{13}c_{11}c_9$ and push them into siding 2 with c_9 as the outermost carriage. We now merge these two 8 carriage trains by pulling c_1, c_3, c_5 and c_7 from siding 1. At this point, the next available carriage in siding 1 is c_2 while the next available carriage in siding 2 is c_9. Since we need c_9 before c_2 in the final train, we switch to siding 2, pull out $c_9, \ldots, c_{15}$, switch back to siding 1 and pull out $c_2, \ldots, c_8$ and finally go back to siding 2 and pull out $c_{10}, \ldots, c_{16}$.

(b) For $n = 1$, we can rearrange any train with $2^1 = 2$ carriages with 1 siding—if the carriages have to be reversed, push the first carriage into the siding, move the second carriage out to the platform and hook up the first carriage before it.

Inductively, assume that with $n-1$ sidings, we can rearrange 2^{n-1} carriages in any order we want. We have to show that with n sidings at our disposal, we can rearrange a train with 2^n carriages. Observe that a train with 2^n carriages can be broken up into 2 trains with 2^{n-1} carriages each. By our inductive assumption for $n-1$, we can use sidings $1, \ldots, n-1$ to rearrange the first half of the train in the the final order that we want and then reverse the train into siding n, thus leaving sidings $1, \ldots, n-1$ free again. We use these $n-1$ sidings to generate (the reverse of) the order that we want for the second half of the train and put it into siding $n-1$. We then merge the two trains of length 2^{n-1} from sidings $n-1$ and n as described in the first part to obtain the rearrangement that we want for the overall train of size 2^n.

Indian National Olympiad in Informatics, 2003

Time: 3 hours *30 April, 2003*

Instructions

(a) You will have to return this question paper at the end of the examination with relevant parts filled out.

(b) There are two questions. You have to write working programs in Pascal, C or C++ to solve each of these questions. You must compile your program yourself and produce an executable file (EXE file).

At the end of each question, there is a space to indicate the location of the source code and the executable file for your solution. Please fill up this information without fail.

(c) All input for your programs will come from the keyboard. All output from your programs should be written to the screen.

Question 1 Chambers in a castle

The floor plan of a castle indicates where the walls are. The outer boundary of the castle is always an unbroken wall. The castle has no doors, only openings in the walls that lead from one room to another. A *chamber* is a collection of rooms that are connected to each other by gaps in the walls. The problem is to:

(a) Count the number of chambers in the castle.

(b) Calculate the area of the largest chamber.

To simplify the problem, the floor of the castle is divided into square tiles. All walls lie at tile boundaries. The area of a chamber is specified in terms of the number of tiles inside the room (without counting the tiles occupied by the surrounding walls).

Figure 1 is an example of a floor plan. Each square is a tile. White tiles represent open space, while black tiles represent walls. In this castle, there are four chambers. The largest chamber has an area of 58.

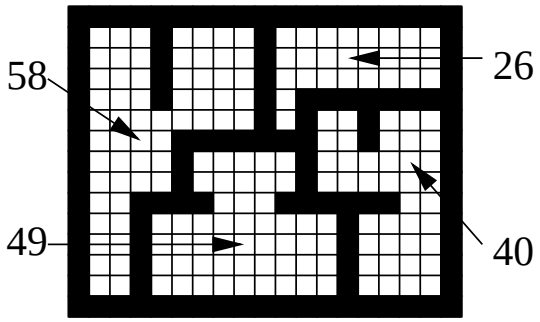

Figure 1

Input format

The first line of input will have two integers `M` and `N`, with $0 < \texttt{M} \leq 500$ and $0 < \texttt{N} \leq 500$. `M` represents the number of rows and `N` the number of columns in the floor plan.

This will be followed by `M` lines of input. Each of these `M` lines will be a sequence of 0's and 1's of length `N`, separated by spaces. A 0 represents an open tile (a white square), while a 1 represents a tile with a wall (a black square). The outer boundary of the castle will always consist of an unbroken wall.

The input corresponding to the floor plan in Figure 1 is given in Figure 2.

```
15 19
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 1 0 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1
1 0 0 0 0 1 1 1 1 1 1 1 0 0 1 0 0 0 1
1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1
1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1
1 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 1
1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Figure 2

Output format

The output of your program should be exactly two lines, each line consisting of a single number. The first line should be the number of chambers. The second line should be the size of the largest chamber.

The correct output for the example in Figures 1 and 2 is shown below.

```
4
58
```

Note: Your program should not print anything other than these two numbers. Please remove all diagnostic print statements before making your final submission. A program with extraneous output will be treated as incorrect!

Question 2 Nearest fraction

Let X be the set of all fractions in reduced form lying strictly below 0 and 1 whose denominator is less than or equal to 99. In other words,

$$\frac{n}{d} \text{ belongs to } X \text{ provided } 0 < \frac{n}{d} < 1 \text{ and } d \leq 99 \text{ and } \gcd(n, d) = 1,$$

where $\gcd(x, y)$ denotes the *greatest common divisor* (or *highest common factor*) of x and y.

For instance, X includes fractions such as 1/3, 11/31 and 24/37 and excludes fractions such as 4/10, 30/70 (both not in reduced form) and 2/101 (denominator too large).

The problem

Given an arbitrary fraction a/b in reduced form whose denominator b is larger than 99, we want to find the pair of fractions in X that are closest to a/b.

In other words, we want to identify fractions u/v and x/y in X such that $u/v < a/b < x/y$ with the property that there is no fraction u'/v' in X such that $u/v < u'/v' < a/b$ and there is no fraction x'/y' in X such that $a/b < x'/y' < x/y$.

For instance, if a/b is 2/101, then $u/v = 1/51$ and $x/y = 1/50$. Here is another example—if $a/b = 322/479$, then $u/v = 41/61$ and $x/y = 39/58$.

Input format

Each test input will consist of a sequence of values a/b for which you have to find the nearest fractions in X. The input is given as follows.

The first line is an integer `M`, $0 < \texttt{M} \leq 500$, indicating the number of fractions in this input test sequence. This is followed by `M` lines of input, each containing a pair of integers `N` and `D` separated by a space, representing the numerator and denominator of the input fraction, respectively. You are guaranteed that $\texttt{D} \geq 100$ and $\gcd(\texttt{N}, \texttt{D}) = 1$.

Here is what the input would look like if the sequence consisted of the two examples 2/101 and 322/479 discussed earlier.

```
2
2 101
322 479
```

Output format

For each input fraction n/d, you have to print out a line containing the values u/v and x/y nearest to n/d in X, where $u/v < n/d < x/y$. Print out u, v, x and y as four integers, in that order, on a single line, separated by spaces. Thus, your output will consist of `M` lines overall, each containing 4 integers.

The correct output for the earlier sample input is shown below.

```
1 51 1 50
41 61 39 58
```

Note: Your program should not print anything other than these `M` lines, each consisting of 4 numbers. Please remove all diagnostic print statements before making your final submission. A program with extraneous output will be treated as incorrect!

Indian National Olympiad in Informatics, 2004

Time: 3 hours *1 May, 2004*

Instructions

(a) You will have to return this question paper at the end of the examination with relevant parts filled out.

(b) There are two questions. You have to write working programs in Pascal, C or C++ to solve each of these questions. You must compile your program yourself and produce an executable file (EXE file).

You must use the GNU set of compilers (`djgpp`). Programs written using Turbo C++ may not run in the evaluation environment and run the risk of not scoring marks.

At the end of each question, there is a space to indicate the location of the source code and the executable file for your solution. Please fill up this information without fail.

(c) All input for your programs will come from the keyboard. All output from your programs should be written to the screen.

(d) Please fill out your contact details below as completely as you can.

Contact details	
Roll Number:	
Name:	
Address:	PIN Code
Phone Number(s): (with STD code)	
Email address:	

Question 1 The Nilgiri Tahr

The *Nilgiri Tahr* is a mountain goat that lives in the Nilgiri hills on the border between Tamil Nadu and Kerala. The tahr lives along steep rock cliffs where grass does not grow easily. The tahr is a very good climber and can balance itself even on the steepest of cliffs. However, since it is a rather short animal, it cannot jump from one rock to another if the difference in height between the rocks is more than one foot.

One such tahr finds itself at the top-left corner of an $M \times N$ rectangular grid of rocks. There is a clump of grass at the opposite (bottom-right) corner. The tahr would like to find a way from its current position to the grass. The tahr cannot jump over rocks and can move from a given rock to only one of its four neighbours (left, right, up or down). The tahr must of course always stay within the grid. Further, remember that it can jump only one foot up or down, so it can move from a rock to its neighbour only if the difference in height between the two rocks is at most one foot. Your task is to help the tahr find such a path to the grass, if it exists. If there is more than one valid path for the tahr to follow, it is sufficient to identify any one path.

For example consider the following grid of rocks, where the number at each position in the grid indicates the height of the rock, in feet, at that position.

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

Here is a path that the tahr can follow to reach the grass. The path follows the rocks labelled by the letters (a), (b), ..., (i), in that order.

7 (a)	9	8	5
6 (b)	3 (e)	2 (f)	4
5 (c)	4 (d)	1 (g)	5
2	6	1 (h)	1 (i)

Input format

The first line of input contains two integers M and N, specifying the number of rows and columns in the grid, respectively, where $1 \leq M \leq 1000$ and $1 \leq N \leq 1000$. The next M lines of input contain N positive integers each. The jth integer on the ith line is the height of the rock in the ith row and jth column of the grid.

Output format

- If there is no path for the tahr to reach the grass, print out a single line containing the number 0.

- If there is a path, the first line of output should be the integer 1. After this, you have to print out any one path that the tahr can follow to reach the grass. The path is described by a sequence of lines. Each line should consist of two integers. The integers X_i and Y_i printed out on line i indicate that the ith step that the tahr takes should be to row X_i and column Y_i.

  Since the tahr always travels from the top-left corner to the bottom-right corner of the grid, the first line of the path is always `1 1` and the last line is always `M N`.

Example

Here is sample input and output corresponding to the example discussed above.

Sample input

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

Sample output

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

Note: Your program should not print anything other than what is specified in the output format. Please remove all diagnostic print statements before making your final submission. A program with extraneous output will be treated as incorrect!

Important

Indicate the location of the source code and executable file for your solution to Question 2 in the boxes below. *If you fail to do this, your solution cannot be evaluated!*

Source file:

Executable file:

Question 2 The Rajah's Wrestlers

Once upon a time, there lived a Rajah who was extremely fond of wrestling. In those days, wrestlers had supernatural powers. To win a match a wrestler relied not only on his own strength but also on a magical ring that he wore while fighting. This ring allowed a wrestler to gain additional strength proportional to the strength of his opponent.

The strength of a wrestler and the magical power of his ring are both positive integers. When wrestler A fights wrestler B, the *fight index* of wrestler A for this match is given by A's own strength plus the strength of wrestler B multiplied by the magical power of A's ring. Each match is won by the wrestler whose fight index is higher for that match.

For example, suppose that wrestler A has strength 10 and wears a ring whose magical power is 3 and wrestler B has strength 18 and wears a ring whose magical power is 4. If these two wrestlers fight each other, wrestler A wins. This is because A's fight index for this match is $10 + (3 \times 18) = 64$ while B's fight index for this match is only $18 + (10 \times 4) = 58$. On the other hand if A faces a wrestler C with strength 15 and a ring whose magical power is 5, then C wins. In this match, A's fight index is $10 + (3 \times 15) = 55$ while C's fight index is $15 + (5 \times 10) = 65$. Similarly, in a match between B and C, C wins.

The Rajah organised a wrestling festival once a year. During this festival, each wrestler fought every other wrestler exactly once. At the end of the contest Rajahh honoured all the wrestlers by inviting them to his court and giving them gold coins.

It was the job of the Minister to decide the order in which the wrestlers got to meet the Rajah. This was an important task, because our Rajah was rather eccentric. He had declared that the number of coins to be given to a wrestler was determined by the number of matches he won *and* his position in the sequence: a wrestler was given one gold coin for each match he won and one gold coin for each wrestler whom he defeated but who was ahead of him in the line to meet the Rajahh.

For instance if the Minister had presented the wrestlers A, B and C described above in the order A, B, C to the Rajahh, then A would get 1 gold coin (for the match he won against B), B would get 0 gold coins (since he won no matches) and C would get 4 gold coins (two for his wins against A and B, one because he beat A and A met the Rajah before him and one more since he beat B and B met the Rajah before him). Instead if the minister had presented them in the order C, A, B then C would get 2 coins (for his wins against A and B), A would get 1 coin (for his win against B) and B would get 0 coins.

The Minister is concerned about the finances of the kingdom and wants to minimise the number of coins handed out by the Rajah to the wrestlers. Your task is to help the Minister decide the sequence in which the wrestlers should be presented to the Rajah so that the number of gold coins handed out is minimum. You are provided with the strengths of all the wrestlers and the magical powers of their rings. You may assume that these values are such that no match results in a tie.

Input format

The first line of input is an integer N indicating the number of wrestlers. Each wrestler is identified by a unique number in the range $\{1, 2, \ldots, N\}$. The next N lines each contain 2 positive integers. For $1 \leq i \leq N$, the first integer on line i+1 denotes the strength S_i of wrestler i and the second integer denotes the magical power R_i of wrestler i's ring. You may assume that $1 \leq N \leq 10000$, and for each wrestler i, $1 \leq S_i \leq 1000$ and $1 \leq R_i \leq 1000$.

Output format

The output consists of N lines, indicating the sequence in which the wrestlers should meet the Rajah to minimise the number of gold coins handed out by the Rajah. Thus, each line of output should be an integer between 1 and N and every integer between 1 and N should appear exactly once in the output. If the integer on line i is j, it means that wrestler j is the ith wrestler to meet the Rajah.

Example

Here is sample input and output corresponding to the example discussed above.

Sample input

```
3
10 3
18 4
15 5
```

Sample output

```
3
1
2
```

Note: Your program should not print anything other than what is specified in the output format. Please remove all diagnostic print statements before making your final submission. A program with extraneous output will be treated as incorrect!

Important

Indicate the location of the source code and executable file for your solution to Question 1 in the boxes below. *If you fail to do this, your solution cannot be evaluated!*

Source file:

Executable file: