acm International Collegiate Programming Contest 2003 IBM event sponsor

ACM International Collegiate Programming Contest 2003

South America

8th November 2003

(This problem set contains eight problems; pages are numbered from 1 to 16)

Hosted by

Universidad de Buenos Aires (UBA), Buenos Aires, Argentina
Universidade Estadual de Campinas (UNICAMP), Campinas, Brazil
Universidade de Fortaleza (UNIFOR), Fortaleza, Brazil
Universidade Federal de Mato Grosso do Sul (UFMS), Campo Grande, Brazil
Universidade Regional Integrada do Alto Uruguai e das Missões(URI), Erechim, Brazil
Universidad del Bío-Bío, Chile
Universidad Dr. Rafael Belloso Chacín, Maracaibo, Venezuela

Problem A

Dice

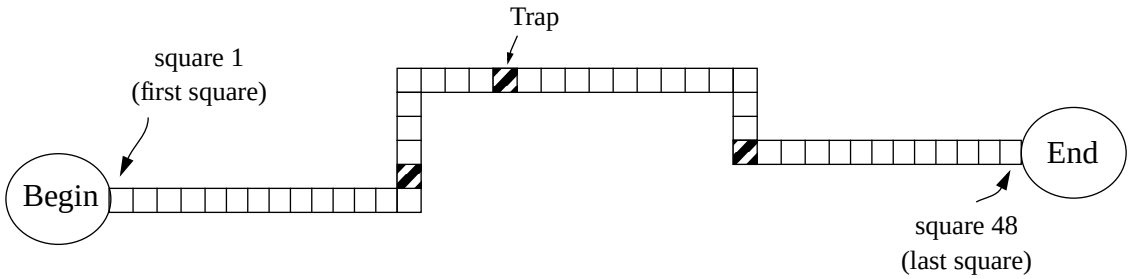
Source file: dice.pas, dice.c, dice.cc or dice.java

A simple boardgame that generations of children have played consists of a board containing a trail of squares and a set of colored pieces. At the beginning of the game each player is assigned a piece; all pieces are initially positioned right *before* the first square of the trail.

The game proceeds in rounds. At each round, players rolls a pair of dice, and move their pieces forward a number of squares equal to the rolled result. Players roll the dice always in the same order (player A, then player B, etc.) in the rounds.

Most of the squares on the board are plain squares, but some are “traps”. If a player’s piece falls on a trap square at the end of the player’s move, the player misses the next round. That is, she/he does not roll the dice, and her/his piece stays one round without moving.

There will be exactly three traps on the trail.



The winner of the game is the player whose piece reaches the end of the trail first. The end of the trail is *after* the last square of the board. Consider, for example, the board in the figure above, which has squares numbered from 1 to 48. At the start, the pieces are positioned at the place marked ‘Begin’ in the figure, that is, before the square number 1. Therefore, if a player rolls a 7 (dice showing 2 and 5 for example), her/his piece is positioned at square number 7 at the end of the first round of the game. Furthermore, if a player’s piece is positioned at square 41, the player needs a roll result of at least 8 to reach the end of the trail and win the game. Notice also that there will be no draw in the game.

You will be given the number of players, the number of squares in the trail, the location of the traps and a list of dice rolls results. You must write a program that determines the winner.

Input

Your program should process several test cases. The first line of a test case contains two integers P and S representing respectively the number of players and the number of squares in the trail ($1 \leq P \leq$

10 and $3 \leq S \leq 10000$). The second line describes the traps, represented by three distinct integers T_1, T_2 and T_3, denoting their positions in the trail ($1 \leq T_1, T_2, T_3 \leq S$). The third line contains a single integer N indicating the number of dice rolls in the test. Each of the following N lines contain two integers D_1 and D_2 ($1 \leq D_1, D_2 \leq 6$), representing the results of the dice rolls. The end of input is indicated by $P = S = 0$. The set of dice roll results in a test will be always the exact number necessary for a player to win the game.

A player is identified by a number from 1 to *P*. Players play in a round in sequential order from 1 to *P*.

The input must be read from standard input.

Output

For each test case in the input, your program should output a single integer: the number representing the winner.

The output must be written to standard output.

Sample input	Output for the sample input
2 10 2 4 8 4 1 1 3 4 1 2 6 5 3 7 4 5 7 7 1 2 2 2 2 1 1 1 1 2 1 1 1 1 0 0	1 3

Problem B

Secure Region

Source file: secure.pas, secure.c, secure.cc or secure.java

You have been hired by Mines Never Again, a non-governmental organization whose aim is to ban the use of landmines. Besides working on political aspects, such as lobbying governments to join the International Campaign to Ban Landmines, MNA also works on disarming mines left by past wars.

Nowadays, mines are detected by satellites or surveillance airplanes. But to disarm a mine you have to get close to it. In most cases, the only way to reach a mined field is by helicopter. To clear the field, you must find the most secure region within the field so that the helicopter can land on it. This region is a rectangle with sides parallel to the coordinate axes, with no mines inside and whose smaller side is the largest possible. More precisely, let A and B be the length of the sides of all possible rectangles that do not contain any mines and $A \leq B$; the most secure region is a rectangle with the largest value of A and the largest value of B. That is, among all rectangles that do not contain any mines and whose smaller side is A (largest possible), the most secure region is a rectangle that has the largest B.

Given the limiting rectangle of a mined field and the positions of all mines inside the field, you must write a program to find the size of the most secure region.

Input

Your program should process data for several mined fields. The first line of a mined field contains four integers X_1, Y_1, X_2 and Y_2 which bound the field. (X_1,Y_1) are the coordinates of the field's lower left corner, (X_2, Y_2) are the coordinates of the field's upper right corner ($-20000 \leq X_1 < X_2 \leq 20000$ and $-20000 \leq Y_1 < Y_2 \leq 20000$). The second line contains a single integer N indicating the number of mines detected in the field ($1 \leq N \leq 300$). The following N lines contain each two integers X and Y describing the position of a mine ($X_1 \leq X \leq X_2$ and $Y_1 \leq Y \leq Y_2$). No two mines have the same location. The end of input is indicated when $X_1 = Y_1 = X_2 = Y_2 = 0$.

The input must be read from standard input.

Output

For each mined field of the input your program should print a line with two integers A and B, where $A \leq B$, describing the size of the most secure region.

The output must be written to standard output.

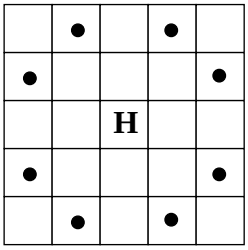
Sample input	Output for the sample input
0 0 100 100 9 0 0 0 100 100 0 100 100 50 50 25 50 50 25 75 50 50 75 -2 0 6 8 3 0 2 2 4 4 6 0 0 0 0	50 50 4 6

Problem C

Runner Pawns

Source file: pawns.pas, pawns.c, pawns.cc or pawns.java

The “Runner Pawns” game is a variant of classic Chess for a single player. It uses a board similar to the chess board, divided in 8x8 squares. As in chess, each square can contain only one piece at a time. The pieces are a number of pawns (the “Runner Pawns”), and a single horse, which is the only piece under command of the player. The objective is to capture all pawns before they get to the last row and become kings.



Possible movements of the horse

Horse moves are said to be in ‘L’ shape, since a horse must always move two squares in one direction and one square in the perpendicular direction. The figure above illustrates horse movements, where the character ‘H’ indicates the horse current position, and the character ‘•’ indicates possible final positions. Notice that in the representation used black and white squares of the chess board are not distinguished.

```
01 02 03 04 05 06 07 08
09 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
```

A board with cells numbered

From position 22, the horse can move to positions 05, 07, 12, 16, 28, 32, 37 or 39. From position 57, the horse can move to positions 42 or 51.

Pawns’ moves are a bit different from chess, since they can only move one square forward, and all of them move at the same time. They never move on a diagonal. Squares of the board are numbered from 1 to 64, as shown above. Pawns move in vertical direction from top to bottom, so that squares numbered 57 to 64 are the pawns’ goal.

Each round of the game is composed by one move of the horse followed by a simultaneous move of all pawns not yet captured.

In order to capture a pawn, the player has to move the horse to a square where a pawn is. A captured pawn leaves the board, and only the remaining ones will move ahead in the next round. To win the game, the player has to capture all pawns. If a pawn gets to the last row, it becomes a king. Then the horse has only one more move to capture it. If it doesn't, the king moves and it means that the game is over and the player loses. Moreover, if the horse moves to a square that will be occupied by a pawn at the next move of the pawns, the horse is captured by the pawn and the player loses.

Your task is to write a program that analyses a "Runner Pawns" diagram and answer whether there is a sequence of movements for the horse to win. If it is possible, your program should determine the minimum number of moves needed by the horse to capture all pawns.

Input

The input contains several instances of the problem, one per line. Each instance starts with an integer P representing the number of pawns ($0 \le P \le 8$), followed by P integers ($1 \le A_1, A_2, \ldots A_P \le 64$) that describe the initial position of each pawn, followed by an integer H ($1 \le H \le 64$) representing the starting position of the horse. The end of input is indicated by a line containing $P = 0$.

The input must be read from standard input.

Output

For each instance of the problem in the input your program must print a single line, containing the answer to the problem. If there is a sequence of moves for the horse to capture all pawns before a surviving king moves (and without the horse being captured by a pawn) then the program should print the length of the minimum sequence of moves that make it possible. Otherwise your program should print the word 'impossible'.

The output must be written to standard output.

Sample input	Output for the sample input
1 1 11 1 60 1 2 33 60 54 0	1 impossible 3

Problem D

The Splitting Club

Source file: club.pas, club.c, club.cc or club.java

The ACM (All Can Meet) club was created with the purpose of attracting people of all ages, with the idea that the people could sit together and share their life experience, to the benefit of all. But as it happened, the club became such a huge success that it was practically impossible to gather all its members at the same place and time. The club then decided to split its members into smaller "sections". In order to make sure that sections are "nice", the director of the club decided to impose the following requirements:

A. all members of the same age should be in the same section,

B. all members should be part of exactly one section,

C. in each section, the maximum number of people with the same age should not be more than R times the minimum number of people with the same age, where R is a rational number between 1.0 and 2.0. The number R is called the splitting factor of the club.

The last condition makes sure that there is no relatively small-age group which might feel uncomfortable in the section. For instance, let denote by [n, m] a group with n members who are m-years old. Then in section {[10, 50], [6, 45], [70, 12], [43, 23]} the maximum number of people with the same age is 70, the minimum number of people with the same age is 6. If $R = 2.0$, then we say this section does not satisfy the requirement (C) since $70/6 > 2.0$. However, we can split this section into two smaller sections, namely {[10, 50],[6, 45]} and {[70, 12], [43, 23]}, which satisfy all the requirements.

Given the splitting factor R and a list of the members of the club, you must write a program to find the minimum number of sections satisfying the three requirements above.

Input

Your program should process several test cases. The first line of a test case contains an integer K and a rational R. K represents the number of different ages in the club ($1 \le K \le 120$), and R represents the splitting factor set by the club director ($1.0 \le R \le 2.0$). The next K lines describe the group members, each line containing two integers N and M, indicating that there are N members who are M-years old in the club ($1 \le N \le 10000$ and $1 \le M \le 120$). The end of input is indicated by a line with $K = 0$ and $R = 0.0$.

The input values will be such that the eventual error in the internal binary representation of R will not affect the result.

The input must be read from standard input.

Output

For each instance of the problem you should output a single line, containing the minimum number of groups satisfying the three requirements above.

The output must be written to standard output.

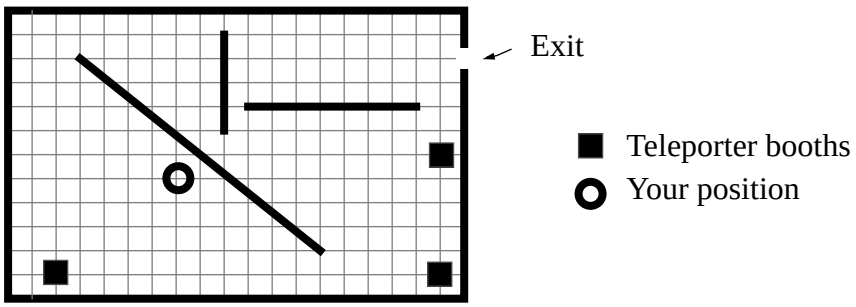
Sample input	Output for the sample input
5 1.7 100 7 18 10 11 17 567 25 62 34 3 1.0 12 18 107 11 250 57 0 0.0	3 3

Problem E

Kryptonite Mine

Source file: krypto.pas, krypto.c, krypto.cc or krypto.java

In the year of 2222, a terrible disaster happened at the kryptonite mine in Mars: a marsquake shook that part of the planet. Differently from earthquakes in Earth, marsquakes are not unusual on Mars. This one, however, caused the mine to start sinking slowly into the soil. The mine has a rectangular external shape, and its interior is like a maze, with high, straight walls and, most importantly, teleporters. Teleporters, as you know, can transport people instantly from one place to another. Teleporters in the mine are old models, using ancient technology, and can only teleport people if there is a clear view from one teleporter booth to another (that is, if there are no obstacles or walls in between the booths). You can see the map of the mine in the figure below.



You are trapped alone inside the mine. Fortunately, you have a map of the whole mine, know your current location, the positions of the walls, the locations of the exit and all teleporter booths. Unfortunately, the marsquake affected the energy system, and you know the teleporters can be used for a limited number of times only.

You want to get out of the mine walking as little as possible, since you sprained your ankle during the marsquake. You must find the route from your present location to the exit that requires the least amount of walking.

Input

The input consists of many test cases. The first line of a test case contains three integers N, M and L, which indicate, respectively, the number of times the teleporters can be used, the number of walls in the mine and the number of teleporter booths ($0 \le N, M, L \le 50$). Each of the next M lines contains four integers X_1, Y_1, X_2 and Y_2, which represent the coordinates of the endpoints of a wall. You may ignore the thickness of walls and assume they do not intersect each other ($-20000 \le X_1 < X_2 \le 20000$ and $-20000 \le Y_1 < Y_2 \le 20000$). The next L lines contain the location of teleporter booths, given by two integers X_p and Y_p. The last line of each test case contains four integers X_b, Y_b, X_e and Y_e where

(X_b, Y_b) are the coordinates of your location and (X_e, Y_e) are the coordinates of the mine's e xit. The end of input is indicated by M=N=L=0.

The input must be read from standard input.

Output

For each test case in the input your program must output a single line, containing an integer representing the distance you need to walk to get out of the mine. Of course, you should not consider the distances you teleported.The distance must be rounded to the nearest integer.

The output must be written to standard output.

Sample input	Output for the sample input
1 1 3 5 -4 5 4 1 0 5 5 9 0 0 0 10 0 1 1 3 5 -4 5 4 0 0 5 5 10 0 0 0 10 0 0 0 0	8 7

Problem F

X-Mart

Source file: xmart.pas, xmart.c, xmart.cc or xmart.java

The well known supermarket chain X-Mart decided to cut costs, reducing the number of different products available in its shops' shelves. The marketing department was concerned that this decision would affect sales, and decided to exploit the reduction of products to promote customer relations.

X-Mart therefore organized an Internet poll, in which customers could choose which products they wanted the supermarket to keep on their shelves, and which products they wanted the supermarket to withdraw from their shelves. The list of currently available products was published on the Internet.

To simplify the polling system, each customer was allowed to choose at most two products to vote for (meaning the supermarket should keep selling it) and at most two products to vote against (meaning the supermarket should stop selling it).

Once the marketing department got all the votes in its database, it wants to know if it is possible to choose a new list of products that pleases ALL voting customers. The marketing department considers that a customer will be pleased when at least one of the products she/he voted for was indeed kept by the supermarket, and at least one of the products she/he voted against was withdrawn from the supermarket's shelves. You may assume a customer does not vote for and against the same product.

Input

Your program should process several test cases. The first line of a test case contains two integers C and P, representing respectively the number of customers and the number of products in the test ($1 \leq C \leq 1000$ and $1 \leq P \leq 10000$). Each of the next C lines describes the preference of one customer, represented as four integers X, Y, S and T ($0 \leq X, Y, S, T \leq P$). X and Y are products the customer wants the supermarket to keep selling, S and T are products the customer wants the supermarket to stop selling. A zero value for any of the variables X, Y, S and T means the customer is not making use of that vote. A line with $C = P = 0$ indicates the end of input.

The input must be read from standard input.

Output

For each test case your program must print one line, containing either the word 'yes' (if it is possible to please all voting customers) or the word 'no' (if it is not possible).

The output must be written to standard output.

Sample input	Output for the sample input
3 4 1 2 3 4 3 4 1 2 2 3 1 4 4 4 1 2 3 4 3 4 1 2 1 3 2 4 1 4 2 3 4 4 1 2 3 4 3 4 1 0 1 3 2 4 2 4 0 3 0 0	yes yes no

Problem G

Telecommunication Partners

Source file: partners.pas, partners.c, partners.cc or partners.java

ICPC, an international telecommunication company, wants to improve its relationship with business subscribers, offering a discount on calls made to a fixed set of telephone numbers selected by the client company. To help ICPC decide on the cost for this new service, they searched their database and produced a list of calls made last year by one company to another. If a company communicated with another company (making or receiving a call) during last year, we will say they are *Business Partners*.

You have been hired by ICPC to process the list of calls from last year and determine the size (in number of companies) of the largest set of companies that are Business Partners of at least K other companies in the same set. That is, you must find a set S of companies such that every company in S has at least K business partners that are also in S (and possibly partners that are outside S), where K is a parameter chosen by the ICPC.

Input

Your program should process several test cases. The first line of a test case contains three integers N, P and K. N represents the total number of companies subscribed to ICPC ($1 \leq N \leq 1000$); companies are identified by numbers between 1 and N. P represents the total number of business partner pairs, produced from last year calls; and K is the minimum number of business partners a company must have in the final set ($1 \leq K \leq N-1$), as described above. The next P lines describe each a business partner pair, represented as two integers X and Y, where X and Y are companies ($1 \leq X \leq N$, $1 \leq Y \leq N$ and $X \neq Y$). The value $N = 0$ indicates the end of input.

The input must be read from standard input.

Output

For each test case from the input, your program should print a single line, containing the size of the largest set of companies found by your program.

The output must be written to standard output.

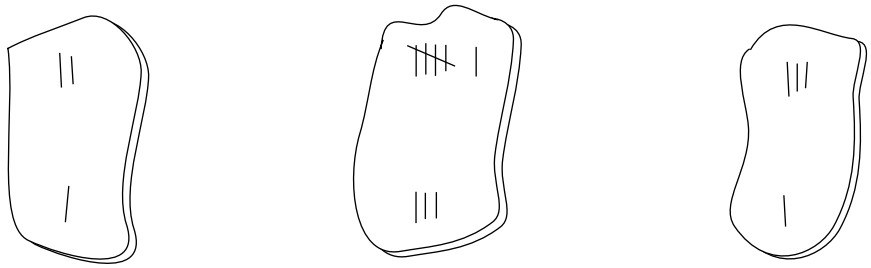
Sample input	Output for the sample input
5 3 1 1 2 4 3 4 5 5 3 2 1 2 4 3 4 5 10 11 2 1 2 1 3 3 2 3 5 5 4 5 6 9 10 8 9 8 7 6 7 6 8 0 0 0	3 0 7

Problem H

Child Play

Source file: child.pas, child.c, child.cc or child.java

Natives from the tiny island of Tookutoo are keen on mathematics, and teach their children to play several math-oriented games. A popular puzzle in Tookutoo is played with ceramic slabs like the ones shown in the figure below.



As it can be seen in the figure above, slabs are similar to dominoes, being divided in two parts; in each part an integer value is imprinted. The slabs above have values [2, 1], [6, 3] and [3, 1]. Note that a slab [*a*, *b*] can also be written as [*b*, *a*].

The puzzle starts with a player receiving a set of slabs chosen randomly from a large and varied pool. Using the given set of slabs, the player has to find an arrangement in which the slabs are put side by side on the table in such a way that the sum of values on the upper side is equal to the sum of values on the lower side. For example, for the set in the figure above, a correct arrangement is

```
1 6 1
2 3 3
```

If it is not possible to find an arrangement using all the slabs chosen, the player may discard one of them, but the value of the sum in the arrangement must be the highest possible. Besides, if more than one slab can be discarded while leaving the same sum, the player must discard the slab [*a*, *b*] such that $a \leq b$ and *a* is the least possible value considering all possible slabs to be discarded.

You must write a program that, given a set of slabs, tries to find an arrangement that satisfies the conditions of the puzzle, discarding one slab if necessary.

Input

Your program should process several test cases. The first line of a test case contains a single integer *N*, the number of slabs in the test ($0 \leq N \leq 400$). Each of the following *N* lines contains two integers X_i and Y_i describing a slab that was given to the player ($0 \leq X_i \leq 1000$ and $0 \leq Y_i \leq 1000$). The value $N = 0$ indicates the end of input.

The input must be read from standard input.

Output

For each test case your program must produce one line describing the result. If it is not possible to find an arrangement, print the word ‘impossible’. If it is possible to find an arrangement, print its sum and the description of the discarded slab (if any). If you had do discard a slab, describe it in the form ‘discard *X Y*’, where $X \leq Y$; otherwise print ‘discard none’.

The output must be written to standard output.

Sample input	Output for the sample input
4 1 4 2 9 2 1 0 4 2 8 1 9 4 3 6 3 1 2 3 1 0	10 discard 1 2 impossible 8 discard none