Departamento de Engenharia Informática

MIUP 2008

MARATONA INTER-UNIVERSITÁRIA DE PROGRAMAÇÃO

Coimbra, 25 de Outubro de 2008

Contents

Team

Scientific Committee

- Filipe Araújo, UC
- Pedro Guerreiro, UAlg
- Pedro Rangel Henriques, UMinho
- Luís Seabra Lopes, UA
- Fábio Marques, UA
- Rui Mendes, UA
- Vasco Miguel Manquinho, IST/UTL
- Arlindo Oliveira, IST/UTL
- Luís Paquete, UC
- Francisco Câmara Pereira, UC
- André Restivo, UP
- Pedro Manuel Pinto Ribeiro, UP
- Fernando Silva, UP
- Paulo Sousa, UL
- Simão Melo e Sousa, UBI
- Delfim Torres, UA

Organizing Committee

- Filipe Araújo
- Marilia Curado
- António Jorge Granjal
- Isabel Lourenço
- Luís Paquete
- Francisco Câmara Pereira

Voluntaries

- Vítor Bernardo
- Rui Costa
- Jorge Farinha
- Alcides Fonseca
- João Gonçalves
- Diana Leite
- Pedro Oliveira
- David Palma
- Ricardo Quintas
- Fernando Rocha
- Cristina Rodrigues
- Hugo Vieira
- Rui Vilão

Program

Place: Departamento de Engenharia Informática, Universidade de Coimbra

When	What	Where
09H00	Reception and Registration	Department's Lobby (entrance)
09H30	MIUP Opening Session	Room B1
09H45	Go to contest rooms	G4.1, G4.3, G4.6 and G5.6
10H00	Contest Beginning	G4.1, G4.3, G4.6 and G5.6
15H00	Contest end	G4.1, G4.3, G4.6 and G5.6
15H30	Problem discussion	Room B1
16H30	Award cerimony and Closing session	Room B1

Practical Aspects

Event format

MIUP follows the general rules of International Collegiate Programming Contest (ICPC) and consists of a 5 hour contest (10 am till 3 pm) with teams of at most three students of Portuguese Universities or Polytechnic Institutions. Each institution can bring up to 4 teams.

All problems are original and were invented by the members of the *Scientific Committee*. The contest environment is Mooshak, an automatic judge developed at University of Porto. It allows for immediate evaluation of submissions, and provides a number of feedback results. The main ones are*Accepted, Presentation Error, Runtime Error, Compile Time Error, Time Limit Exceeded, Memory Limit Exceeded, Wrong Answer and Invalid Function*. Programming languages accepted are C, C++, Java and Pascal.

During the contest, there is online support from the side of the *judges* (members of the Scientiic Committee), and the classification is based on the number of problems solved. In case of a draw, Mooshak sums all the times needed for solving the problems (with 20 min. penalty for every non-Accepted submission) and places first the teams that needed less time.

Documentation

During the contest, documentation will be available at http://miup2008.dei.uc.pt/docs/. Aside from the Mooshak platform, this is the only web link accessible for contestants.

Compilation

Constraints

- Maximum compilation time: 60 seconds
- Maximum source code size: 100 kb
- Every source code must be submitted in a single file
- In case of Java submissions, the ".java" file has to have the same name as the class that contains the main method. There is no limit for the number of classes to be contained in that file.

Command line

- gcc -Wall -Wno-unused -lm

- g++ -Wall -Wno-unused
- fpc -v0w
- javac

Versions

- gcc 4.1.2
- Free Pascal 2.2.0
- jdk 1.6.0_06

Available IDEs

All machines will run Fedora Core 8 linux, with the following IDE/editors:

- Fedora Eclipse 3.3.0
- Xemacs 21.5
- Gnu emacs 22.1.1
- Vim 7.1.135
- Gedit 2.20.3
- Kate 2.5.8
- Kdevelop 3.5.0-4
- DDD 3.3.11
- Netbeans 6.1
- Kwrite 4.5.8

Runtime Constraints

- Max CPU time: 1 second (except when stated otherwise)
- Available memory for dynamic and global variables: 32 MB
- Available memory for local variables and execution stack: 2 MB

Documentation

Online Help

Contestants have three sources of online help:

- Mooshak automatic help. Press the "help" button when you're logged in. This is a page that has information on available (memory and time) limits, compilers and a description of a number of Mooshak items (e.g. explanation of what a "Presentation Error" means).
- Mooshak online judge help (Questions). Press the "Ask" button and you can ask a question directly to the judge. Don't forget to choose the correct problem (otherwise the judge won't know what your question refers to). And please check before in the "Questions" section if there is information that already satisfies your doubts.
- STL, Javadocs and Free Pascal docs. Just access `http://miup2008.dei.uc.pt/docs/`.

Contest Material

As usual, for our contest, we follow the ICPC SWERC's Rules, so the teams are not allowed to bring any reference materials such as books, program listings or notes. No electronic device or machine readable media (including cell phones, digital cameras, mp3 players, USB Pens) are allowed. You may bring pens or pencils, but not paper (it will be provided).

Your team should prepare a notebook consisting of no more than 25 printed A4 pages (note that this is different from 25 A4 sheets printed in both sides, which would correspond to 50 pages), font size greater or equal to 8. Each team member may use an exact copy of the notebook. You have to submit your team notebook at the registration desk on arrival. The team notebook (provided by you) will be available during the contest (Note: excerpt from SWERC rule page).

Problems

Problem A – Hasse Diagram

Introduction

Consider a partially ordered set (poset, for short) (A,⊆), where A is a set and ⊆ a partial order. An Hasse Diagram is a very convenient and compact graphical representation of a partially ordered set displayed via the cover relation of the poset. In a few words, the cover relation of a relation R is R minus the transitivity and the reflexivity.

Let us explain now how to draw a Hasse Diagram. Each node of the diagram is an element of the poset, and if two elements x and y are connected by a line then $x \subseteq y$ or $y \subseteq x$. The position of the elements and the connections are drawn respecting the following rules:

1. If $x \subseteq y$ in the poset, then the point corresponding to x appears below the point corresponding to y.
2. The transitivity of the poset is graphically omitted, that is, if $x \subseteq y$ and $y \subseteq z$, then, by transitivity of the partial order $\subseteq$, $x \subseteq z$. In this case the connection x-z is omitted.
3. Along the same lines, the reflexivity is grafically omitted. In other words, for all x in the graph, the connection x-x is removed.

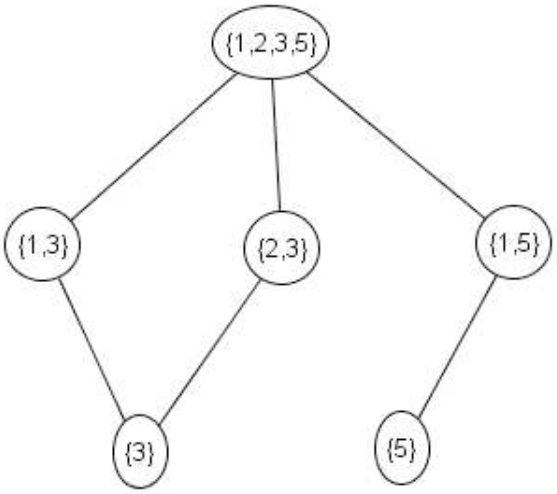


Figure 1: Hasse Diagram representation of the poset $(\mathcal{S} = \{\{1,2,3,5\},\{2,3\},\{5\},\{3\},\{1,3\},\{1,5\}\},\subseteq)$

Problem

Given a set Q of naturals such that $|Q| = \boldsymbol{P}$. Consider a set S of $\boldsymbol{N}$ subsets of Q, that is, $S \subseteq \mathcal{P}(Q)$ and $|S| = \boldsymbol{N}$. $(S, \subseteq)$ is then a partially ordered set. Each element K of S is a set on its own, with a maximum of $\boldsymbol{P}$ elements. The elements of K are positive and are increasingly ordered.

Given such a poset, your goal is to compute all the possible paths in its Hasse diagram from the highest elements to the lowest ones.

Constraints

- Q and therefore the elements of S only contain values that fit a 16 bits representation.
- $0 < \boldsymbol{N} < 100$
- $0 < \boldsymbol{P} <= 100$

Input

The first line of the input contains one positive integer number $\boldsymbol{N}$, which is the number of elements of S. The following $\boldsymbol{N}$ lines contain the elements of S, one for each line. In each integer sequence, the elements are separated by a single space.

Output

Each line of the output file has the nodes of a path, of the Hasse Diagram, separated by `->`. The paths must be lexicographically ordered. Observe Figure 1 and the sample output.

Sample Input

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

Sample Output

```
1 2 3 5->1 3->3
1 2 3 5->1 5->5
1 2 3 5->2 3->3
```

Problem B – The Toxic Clouds of Proxima III

Introduction

Rodericus is an intrepid adventurer (a little too temerarious for his own good) who boasts about exploring every corner of the universe, no matter how inhospitable. In fact, he doesn't visit the planets where people can easily live in, he prefers those where only a madman would go with a very good reason (several millions of credits for instance). His latest exploit is trying to survive in Proxima III. The problem is that Proxima III suffers from storms of highly corrosive acids that destroy everything, including spacesuits that were especially designed to withstand corrosion.

Our intrepid explorer was caught in a rectangular area in the middle of one of these storms. Fortunately, he has an instrument that is capable of measuring the exact concentration of acid on each sector and how much damage it does to his spacesuit. Now, he only needs to find out if he can escape the storm.

Problem

The problem consists of finding an escape route that will allow Rodericus to escape the noxious storm. You are given the initial energy of the spacesuit, the size of the rectangular area and the damage that the spacesuit will suffer while standing in each sector.

Your task is to find the exit sector, the number of steps necessary to reach it and the amount of energy his suit will have when he leaves the rectangular area. The escape route chosen should be the safest one (i.e., the one where his spacesuit will be the least damaged). Notice that Rodericus will perish if the energy of his suit reaches zero.

In case there are more than one possible solutions, choose the one that uses the least number of steps. If there are at least two sectors with the same number of steps (X_1, Y_1) and (X_2, Y_2) then choose the first if $X_1 < X_2$ or if $X_1 = X_2$ and $Y_1 < Y_2$.

Constraints

$0 < E \leq 30000$	the suit's starting energy
$0 \leq W \leq 500$	the rectangle's width
$0 \leq H \leq 500$	rectangle's height
$0 < X < W$	the starting X position
$0 < Y < H$	the starting Y position
$0 \leq D \leq 10000$	the damage sustained in each sector

Input

The first number given is the number of test cases. Each case will consist of a line with the integers E, X and Y. The following line will have the integers W and H. The following lines will hold the matrix containing the damage D the spacesuit will suffer whilst in the corresponding sector. Notice that, as is often the case for computer geeks, (1,1) corresponds to the upper left corner.

Output

If there is a solution, the output will be the remaining energy, the exit sector's X and Y coordinates and the number of steps of the route that will lead Rodericus to safety. In case there is no solution, the phrase `Goodbye cruel world!` will be written.

Sample Input

```
3
40 3 3
7 8
12 11 12 11  3 12 12
12 11 11 12  2  1 13
11 11 12  2 13  2 14
10 11 13  3  2  1 12
10 11 13 13 11 12 13
12 12 11 13 11 13 12
13 12 12 11 11 11 11
13 13 10 10 13 11 12
8 3 4
7 6
4  3  3  2  2  3  2
2  5  2  2  2  3  3
2  1  2  2  3  2  2
4  3  3  2  2  4  1
3  1  4  3  2  3  1
2  2  3  3  0  3  4
10 3 4
7 6
3  3  1  2  2  1  0
2  2  2  4  2  2  5
2  2  1  3  0  2  2
2  2  1  3  3  4  2
3  4  4  3  1  1  3
1  2  2  4  2  2  1
```

Sample Output

```
12 5 1 8
Goodbye cruel world!
5 1 4 2
```

Problem C – Where could we meet?

Introduction

George is the CEO of a chain of stores. From time to time, George wants to arrange meetings with workers from any store. He is not a fan of the new technologies and prefers to talk directly with people.

These meetings always take place at the end of the work day. George realized that due to rush hour, it is impossible to go from some stores to the management building and get there on time.

Since each store also has a meeting room for the store manager to use, George decided that his meetings could also take place at one of the stores. However, workers from any store should be able to get to the meeting place on time during rush hour.

Problem

Consider that you are given a set of directed connections between stores. These connections represent roads that never have rush hour problems at the end of the day. A store s can be used to hold the meeting between George and the workers if there is a path (using one or more connections) from all other stores to s. Your task is to determine how many stores George can use to meet with the workers.

Input

The input is a directed map specified as follows:

- One line with two integers N and M, where N represents the number of stores and M the number of available directed connections between stores.
- A list of M lines where each line contains two integers s_i and s_j (with $1 \leq s_i \leq N$ and $1 \leq s_j \leq N$) that represent an available connection from store s_i to store s_j.

Output

The output is a single integer with the number of stores from which there is a path from all other stores.

Constraints

$0 < N \leq 10^6$
$0 < M \leq 10^6$

Sample Input 1

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

Sample Output 1

```
2
```

Sample Input 2

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

Sample Output 2

```
0
```

Problem D – The Magician's Problem

Introduction

David is a well-known Magician for crossing concrete walls of buildings during his shows. However, he is losing his ability every time he has to go through large concrete walls. To overcome this problem, David is now retrieving satellite data of the building before going into action. This data consists of a three-dimensional description of the building concrete structure. He uses this information to find the set of walls that minimize the amount of concrete to go through in a straight line. You should write an algorithm that helps David to choose the right walls to cross. Obviously, the algorithm should provide the answer *on-the-fly*, even for sky-wrappers with thousands of floors...

Problem

Given a set of three-dimensional orthogonal boxes (the walls), choose a non-empty subset such that the sum of the width of the chosen boxes, measured in a horizontal line segment (i.e. parallel to the x-coordinate), is minimized.

The building has several floors. There may be several overlapping boxes in each floor. In this case, the width of the overlapping should be subtracted from the total width of the overlapped boxes. For instance, given two boxes, A and B, with $\texttt{width}(A) = 10$, $\texttt{width}(B) = 15$, and $\texttt{width}(A \cap B) = 5$, then $\texttt{width}(A \cup B) = 20$ $(10 + 15 - 5)$.

Important Note: David is only allowed to cross at fixed half-integer y-coordinate values (that is, 0.5, 1.5, 2.5, ...) and within the limits of each floor.

Input

Each test file starts with the number of boxes, followed by the location of each box, with the following sintaxe: `<min x> <min y> <max x> <max y> <min floor> <height>`.

The numbers `<min x>` and `<min y>` are the minimal x- and y-coordinates of the box in the bidimensional plane, while `<max x>` and `<max y>` are the maximal x- and y-coordinates. The number `<min floor>` is the floor number where the base of the box is located. The number `<height>` is the height of the box in terms of number of floors. Note that a `<height>` equals to zero means that the box is located in a single floor. The coordinates are nonnegative integers. Do not expect any kind of structure on the location of boxes in the building.

Output

Total width of the chosen boxes (minus the width of the overlapping).

Constraints

- Number of boxes: $[9, 200]$
- Range of x and y coordinates: $[0, 15000]$
- Maximum number of floors: 40000

Sample Input

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

Sample Output

```
2
```

Note: See the solution for the test above in Figure 2. The dashed arrow indicates the direction that David should follow.

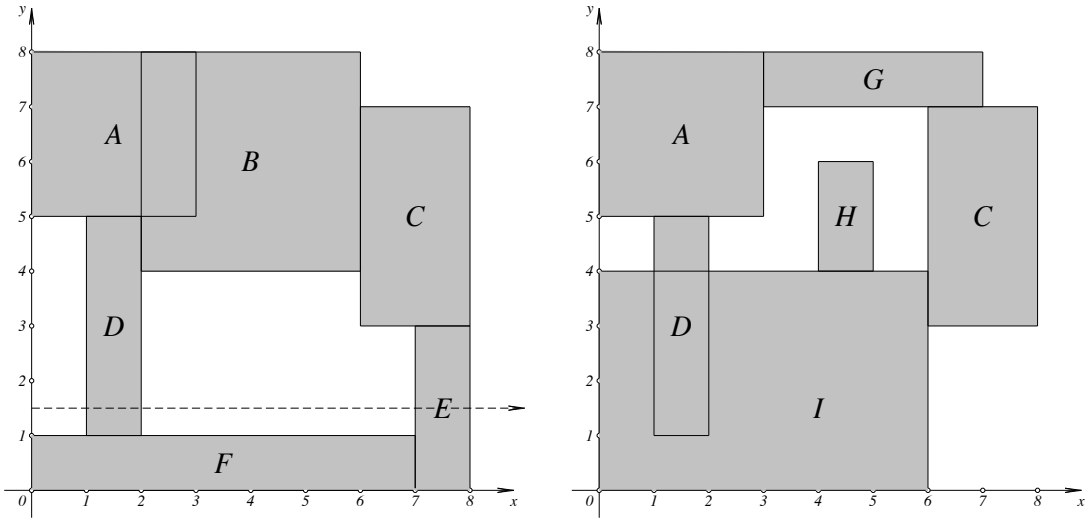


Figure 2: Ground floor (left plot) and first floor (right plot).

Problem E – Tile Game

Problem

John has bought a new game that is driving him crazy. The game proposes simple puzzles where the player has to find a way to transform a tilled square into another tilled square pressing its tiles.

Each tile can be either white or black and each puzzle has an initial configuration, a goal configuration and a set of rules. These rules allow users to change tiles from black to white and vice-versa.

The rules are also very simple. Each tile has a different transformation rule. By clicking a tile, its transformation rule is applied to the whole puzzle. A transformation rule states, for each tile, if it remains unchanged (X), changes to either white (W) or black (B) or if it flips (F) (see Figure 3). Flipping means that the tile becomes white if it was black and black if it was white.

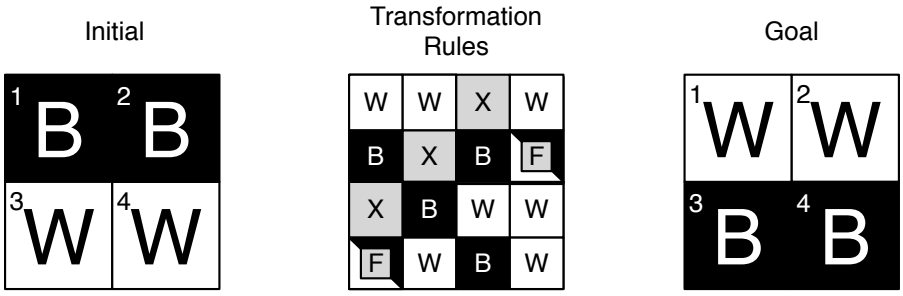


Figure 3: Example Game

A solution to a certain puzzle is the set of tiles that the player needs to press in order to go from the initial state to the goal state. The lower the number of tile presses the user takes the better the score. Figure 4 represents a possible solution for the puzzle just presented..

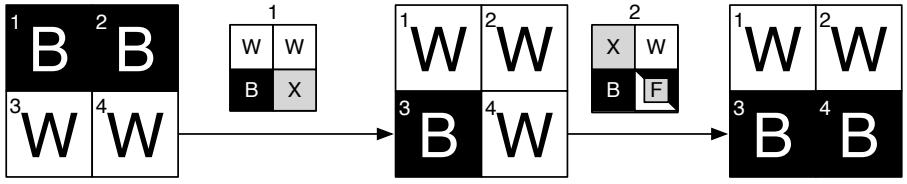


Figure 4: Example Solution

John is desperate to beat his younger brother high-scores and he asked for your help. Your task is to develop a program that given a puzzle of size $\boldsymbol{D}$ outputs the fastest way to solve it.

Constraints

$1 \leq \boldsymbol{D} \leq 3$

Input

Each input file will contain the description of a puzzle with the first line containing a single integer representing its dimension $\boldsymbol{D}$.

The next $\boldsymbol{D}$ lines will contain the initial configuration of the puzzle. Each one of these lines will be composed of $\boldsymbol{D}$ characters with the values B or W representing the type of each tile.

The next $\boldsymbol{D}$ lines will contain the expected final result of the puzzle using the same notation as for the initial configuration.

The remaining of the file will contain the puzzle rules. These will use $\boldsymbol{D} \times \boldsymbol{D}$ lines ($\boldsymbol{D}$ lines for each one of the $\boldsymbol{D}$ transformation rules). For each tile there will be $\boldsymbol{D}$ lines containing its transformation rules.

Each transformation rule will have $\boldsymbol{D}$ lines. Each one of these lines will be composed of d characters with the values X, F, B, or W (Keep unchanged, flip, change to black or change to white).

Output

The output will contain one single line containing the optimum solution to the puzzle. The optimum solution is the one that uses less tile presses. If more than one optimum solution exists the program should output the solution that comes lexicographically first.

The solution should be printed as a sequence of space separated numbers from 1 to $\boldsymbol{D} \times \boldsymbol{D}$ representing the tiles to be clicked. With tile number 1 being the one at the top left and tile number$\boldsymbol{D} \times \boldsymbol{D}$ being the one at the bottom right.

If no solution exists the program should output a single line containing "No solution".

If the initial and final configuration are the same the program should output a single line containing "No moves required".

Sample Input 1

```
2
BB
WW
WW
BB
WW
BX
XW
BF
XB
FW
WW
BW
```

Sample Output 1

```
1 2
```

Sample Input 2

```
2
BB
WW
WW
```

```
BB
WW
BX
XW
BW
XB
FW
WW
BW
```

Sample Output 2

```
No solution
```

Problem F – Finding Loops

Introduction

The YouTrace Collaborative Map Generator receives GPS traces or trajectories, sent by whoever wants to contribute, and integrates them in a "Map of the World". A GPS trace is a sequence of GPS (latitude, longitude) points. The idea is that, with sufficient contributions, one gets a quite complete map of the world, for free. Colaboratively!

Well, the problem is a bit harder than it may seem at a first sight, and has many little hurdles. One of which is finding loops in incoming traces. We call a *loop* whenever a trace draws a closed shape with its own GPS positions. To clarify, let us assume that a GPS Trace T consists of the sequence of points $X_1, X_2, X_3, ..., X_n$. Let us also assume that two points *coincide* whenever their distance is less than a range $\boldsymbol{R}$. There is a loop from i to j, with $i < j$ and $i, j \in T$, when X_i coincides with X_j and there is no point X_k coinciding with either X_i and X_j, such that $i < k < j$. In other words, there is a loop that starts in i and ends in j.

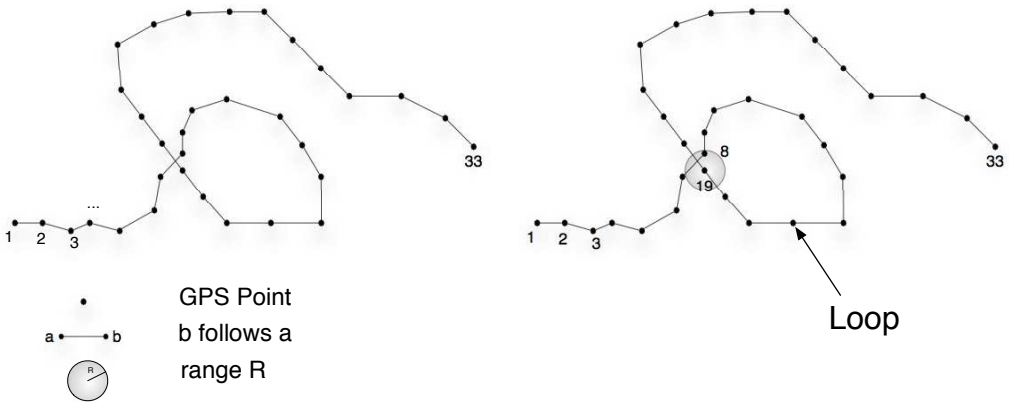


Figure 5: Left: input trace; Right: loop found from 8 to 19.

In Figure 5, we can see an example of a loop found in a GPS trace. Notice also that some tolerance must be given such that no successive points fall in the range $\boldsymbol{R}$. In other words, we *do not* consider points that fall within a "time" window $\boldsymbol{W}$. So formally, this means that the condition $j - i > \boldsymbol{W}$ has to be met.

Another issue is that, whenever a loop from i to j is found, every point so far no longer counts, as if the positions from the beginning to j have simply "disappeared" and will not be counted for in the remaining processing. In Figure 6, we give an example of a trace with 52 points, with $\boldsymbol{W}$=5. We find two loops (from 8 to 19 and from 29 to 44). Notice that position 49 doesn't make any loop with 48 or 50 because they are within the window $\boldsymbol{W}$.

Problem

Your task is to make a program that, given as input the range $\boldsymbol{R}$, the window $\boldsymbol{W}$ and a trace of $\boldsymbol{N}$ points, identify the loops in that trace.

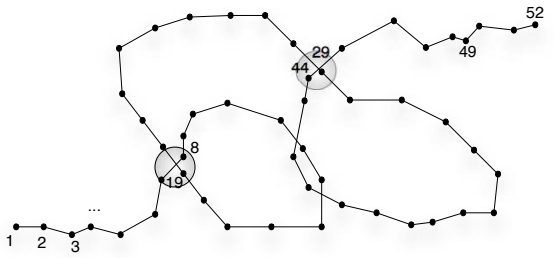


Figure 6: Two loops, assuming $\boldsymbol{W}$=5.

Input

The first line of input contains three positive values: $\boldsymbol{N}$, $\boldsymbol{R}$ and $\boldsymbol{W}$. $\boldsymbol{N}$ and $\boldsymbol{W}$ are integers, $\boldsymbol{R}$ is a real number that indicates the range in meters. Then, follow $\boldsymbol{N}$ pairs of real numbers, px and py, corresponding to the coordinates of each point. Notice that, for simplicity, px and py correspond to meters in an euclidean space (rather than the standard latitude/longitude measures).

Output

For each loop your program should print the values of i and j in a separate line, in the order that they happen in the trace.

Constraints

- $0 < \boldsymbol{N} <= 100000$
- $0 <= \boldsymbol{R} <= 200$
- $0 < \boldsymbol{W} <= 100$

Sample Input

```
21 10.0 10
49.99 3.3
50.2 3.2
55.55 5.45
65 7.02
70.2 7.55
75.2 10.3
80.34 10.7
90.6 11.3
89.9 12.7
92.3 12.3
85.2 11.89
92.2 10.7
92.23 8.45
80.34 8.04
75.98 8.04
```

```
70.88 8.78
60.45 9.7
50.34 11.34
46.2 10.3
40.1 11.3
38.3 10.7
```

Sample Output

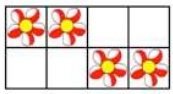
```
4 16
```

Problem G – School Garden

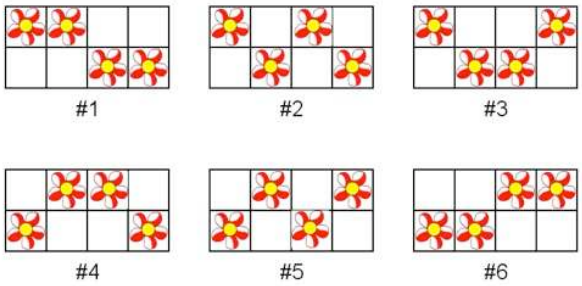
Introduction

The school has a new garden and you are in charge of planting the flowers. The garden can be seen as a rectangular grid of $\boldsymbol{R}$ rows and $\boldsymbol{C}$ columns, in which each grid cell has only space to accommodate one flower. Due to aesthetical reasons, your orders are to plant the flowers in such a way that each grid row has the exact same amount of $\boldsymbol{F_R}$ flowers and columns also have an equal number of $\boldsymbol{F_C}$ flowers among themselves.

To give a practical example, imagine a case where $\boldsymbol{R} = 2$, $\boldsymbol{C} = 4$, $\boldsymbol{F_R} = 2$ and $\boldsymbol{F_C} = 1$ (garden with 2 rows and 4 columns; 2 flowers per row and 1 flower per column). A possible placement of the flowers would be:



However, there are several other ways of putting the flowers, respecting the same requirements. In particular, for the same request, there would be 6 different ways of placing the flowers, as depicted on the following figure:



You got curious about the many possibilities in placing the flowers and tried to count how many possibilities were there for the general case.

Problem

Given the dimensions $\boldsymbol{R}$ and $\boldsymbol{C}$ of a grid (number of rows and columns) and the numbers $\boldsymbol{F_R}$ and $\boldsymbol{F_C}$ (number of flowers in each row and in each column), you must calculate in how many possible $\boldsymbol{W}$ ways you can position the flowers, such that each row and column have the desired number of flowers. Each test file will have $\boldsymbol{T}$ different test cases.

Constraints

$1 <= \boldsymbol{T} <= 10$	Number of test cases per file
$1 <= \boldsymbol{R}, \boldsymbol{C} <= 10$	Number of rows and columns
$1 <= \boldsymbol{F_R}, \boldsymbol{F_C} <= 7$	Number of flowers in each row and column
$1 <= \boldsymbol{W} < 2^{31}$	Number of ways to position the flowers

Input

The first line of input has a single integer $\boldsymbol{T}$, indicating the number of test cases.

Then follow exactly $\boldsymbol{T}$ lines, each one of them detailing one test case. Each line has four integers $\boldsymbol{R}$ $\boldsymbol{F_R}$ $\boldsymbol{C}$ $\boldsymbol{F_C}$ separated by single spaces. The numbers always come in that order. $\boldsymbol{R}$ indicates the number of rows of the grid, $\boldsymbol{F_R}$ indicates the number of flowers each row must have, $\boldsymbol{C}$ indicates the number of columns and $\boldsymbol{F_C}$ indicates the number of flowers each column must have. All test cases will have at least one valid arrangment of the flowers.

The example in the figure above is the first test case in the example input.

Output

The output should consist of exactly $\boldsymbol{T}$ lines, each one with a single integer $\boldsymbol{W}$ indicating how many ways are there to place the flowers respecting the requirements for the respective input. Note that these lines should come in the same order as the test cases appear in the input. Rest assured that in the test cases given the number of ways will always be smaller than 2^{31}.

Sample Input

```
4
2 2 4 1
2 1 2 1
4 2 4 2
2 5 10 1
```

Sample Output

```
6
2
90
252
```

Problem H – Heads and Tails

Introduction

Ana likes to spend her time inventing new games with her collection of coins. This time, she has set out four coins, heads up. In the first move, she turns over the first coin. In the second move, she turns over the first two coins; in the third move, she turns over the first three coins; the fourth move, the first four coins. On the fifth move, she starts all over again, turning over the first coin, and so on.

Starting configuration:	H	H	H	H
1st move:	T	H	H	H
2nd move:	H	T	H	H
3rd move:	T	H	T	H
⋮	⋮	⋮	⋮	⋮

What are the sides of the four coins after 1995 moves?

Problem

Given a starting configuration of N coins, what are the sides of the N coins after k moves, if in the first move we turn over the first coin; in the second move we turn over the first two coins; ...; in the Nth move we turn over all the N coins; on the $(N + 1)$th move we start all over again, turning over the first coin; etc.

Input

The input has three lines: the 1st line with the number N of coins; the second line with the starting configuration – a sequence of N H's and T's separated by a space; the 3rd line with the number k of moves.

Output

The output must have one line with the kth move: the sequence of N H's and T's, separated by a space, that is obtained from the starting configuration after k moves.

Constraints

$1000 < k < 1000000000$

Sample Input

```
4
H H H H
1995
```

Sample Output

```
T H T H
```

Problem I – Dream Girl

Introduction

John is still living with his parents, at 35, enjoying a live free from mundane worries, such as paying electricity bills, buying his own food, doing his laundry or, worse of all, taking care of children. He is a popular young man (well, not so young any more...), his friends love his company, he is known to have had a string of girlfriends, never committing, jumping out before things became too serious. Well, the times they are a-changing, and, keeping it secret, John decided to get married. Knowing by experience that the world is full of wonderful girls, and wanting to get the most wonderful of them all for a wife, he designed a clever plan to get "the one". It is as follows.

John will date a new girl every 6 months, for the next 5 years, bringing him to 40 years of age and 10 more girls in his curriculum. He will carefully note down each one's pluses and minuses, and by the end of the semester he will explain to her candidly that after all perhaps they were not made for each other, and then pass to the next girl. After 5 years, he will be able to rank all the 10 women, and then he will go back to the one in top position and contritely ask her back and propose marriage. This is a flawless plan, except for a small detail: it is unlikely that all girls are so much in love with John that they just sit there waiting that he returns. The only way to be sure that the girl is available is for John to propose while they are still dating. Otherwise, the girl, who is not stupid (John never dates stupid girls), will have moved on also, now going out, or having married, a guy who is a better lover that John is, even if this is inconceivable to him. So, John tweaked his plan. He will date some girls, in an "initial sample", for instance 3 girls, according to the original plan. Then he will continue and propose immediately to the first one that ranks better than all the first 3. If he never finds one, he will settle for the last girl. The only question still worrying John is the effectiveness of his plan. In other words, what is the mathematical probability of him getting his dream girl using the scheme he devised? And, if he is a bit less ambitious, what is the probability of him getting not necessarily the best girl of all but one of the top 2 girls, or more generally, one of the top $\boldsymbol{T}$ girls, for some $\boldsymbol{T}$ in his mind?

Problem

Your task is to write a program that, given the number of girls John is willing to date, $\boldsymbol{N}$, and the number of girls in the initial sample, $\boldsymbol{S}$, computes the probability that John will get one of the top $\boldsymbol{T}$ girls.

Input

The input file contains only one line, with three integers, representing the values of $\boldsymbol{N}$, $\boldsymbol{S}$ and $\boldsymbol{T}$. Each two are separated by a single space.

Output

The output file will have only one line, in which there is an integer number, expressing the probability in percent that with the described scheme, for $\boldsymbol{N}$ girls, with an initial sample of $\boldsymbol{S}$ girls, John will get one of the top $\boldsymbol{T}$ girls in his ranking.

You are only required to provide an approximation of the probability, with an absolute error less or equal to 1. For example, if the jury's value is 57, then 56, 57 or 58 will be accepted.

Constraints

$0 < N <= 100$	N	The number of girls
$0 <= S < N$	S	The size of the initial sample
$1 <= T <= N$	T	John will settle for one of the top T girls

Sample Input 1

```
4 2 1
```

Sample Output 1

```
42
```

Explanation: John decides to date four girls and propose marriage to the first one he meets after the second who is better than any of the first two girls, or to the last girl, in none of the remaining girls are better than any of the first two. The probability that girl he proposes to is the best girl of all is 5/12, which rounds to 42%.

Sample Input 2

```
10 9 1
```

Sample Output 2

```
10
```

Explanation: In this case, John decides to date ten girls and propose marriage to the tenth. In fact, the plan is to propose the first girl he meets after the ninth, which is better than all the first nine, or to the last girl. It turns out that he will propose to the tenth. The probability that the tenth girl is the best girl of all is 10%.

Sample Input 3

```
5 0 4
```

Sample Output 3

```
80
```

Explanation: There are five girls and John decides to propose to the first one. The probability that the first girl is one of the top four is 80%. Note that the first girl is the first girl that is better than any of the zero first girls. Indeed, the size of the initial sample can be zero.