The Pilots Brothers' refrigerator

Task code: PILOTS Time limit: 1 sec

The game "The Pilots Brothers: following the stripy elephant" has a quest where a player needs to open a refrigerator.

There are 16 handles on the refrigerator door. Every handle can be in one of two states: open or closed. The refrigerator is open only when all handles are open. The handles are represented as a matrix 4x4. You can change the state of a handle in any location **[i, j]** ($1 \le$ **i, j** ≤ 4). However, this also changes states of all handles in row **i** and all handles in column **j**.

The task is to determine the minimum number of handle switching necessary to open the refrigerator.

The input file PILOTS.IN contains four lines. Each of the four lines contains four characters describing the initial state of appropriate handles. A symbol "+" means that the handle is in closed state, whereas the symbol "-" means - "open". At least one of the handles is initially closed.

The output file is PILOTS.OUT. The first line of the file contains N – the minimum number of switching. The rest N lines describe switching sequence. Each of the lines contains a row number and a column number of the matrix separated by one or more spaces. If there are several solutions, you may give any one of them.

Example of input data

```
-+--
----
----
-+--
```

Example of output data

```
6
1 1
1 3
1 4
4 1
4 3
4 4
```

A safe way

Task code: SAFEWAY Time limit: 3 sec

There is a map of a minefield. The border of the minefield is a polygon which has not got self-intersections or self-contacts. There are two points **A** and **B** outside of this minefield or on its border. You need to find the minimum length way from **A** to **B**. Obviously the way can not cross the minefield. However, it may contact edges or vertices of the bounding polygon.

The input file SAFEWAY.IN consists of several lines. The first line contains four numbers $\mathbf{X_A}$, $\mathbf{Y_A}$, $\mathbf{X_B}$ and $\mathbf{Y_B}$ – the coordinates of two given points. The second line contains integer number **N**, $3 \le$ **N** ≤ 100 – the number of vertices of the bounding polygon. Finally, each of the next **N** lines contains a coordinates **X** and **Y** of one polygon vertex, in the counterclockwise order. All the coordinates are integer numbers between –100000 and 100000. The numbers are separated by one or more spaces.

The output file SAFEWAY.OUT consists of one line containing the length of a shortest safe way, with the accuracy 0.0001.

Example of input data

```
0 0 5 5
4
1 0
3 0
3 2
1 2
```

Example of output data

```
7.2361
```

Triangles

Task code: TRIANGLE Time limit: 1 sec

triDesign company produces different logical games and puzzles for children. One of the games called *triSuper* is basically a set of sticks. The length of a stick is measured in millimeters and some of sticks in a set may be of the same length.

Authors of the game think for some reason that a child being given a *triSuper* game set uses the sticks to construct triangles. Doing so, the child will eventually realize that it is not always possible to construct a triangle from any three sticks. This is the educational value of the game – to study "the inequality of a triangle".

A particular feature of the game is that each game set is unique. Furthermore, each game set is tested after production. The game set is rejected if it breaks any of the following rules.

1. None three sticks from the set can be used to construct a triangle.
2. Any three sticks from the set can be used to construct a triangle.

As far as a game set may contain a lot of sticks, it is necessary to develop special program to help testing game sets. This is what you need to do.

The input data are in the text file TRIANGLE.IN. It describes one game set. The first line of the file contains an integer number **N** ($1 \le N \le 1\,000\,000$). The second line contains **N** integer numbers $\mathbf{A_1}$, $\mathbf{A_2}$, …, $\mathbf{A_N}$, separated by spaces ($1 \le \mathbf{A_i} \le 2\,000\,000\,000$). $\mathbf{A_i}$ is the length of the stick number **i** in the set. Sticks in the set a so thin, that you can disregard their thickness.

The output text file TRIANGLE.OUT has to contain a single line. If a game set is rejected than the line is "**The set is rejected.**". Otherwise, the line is "**The set is accepted.**"

Example of input data #1	Example of output data #1
3 1 2 3	The set is rejected.

Example of input data #2	Example of output data #2
4 4 4 4 4	The set is rejected.

Example of input data #3	Example of output data #3
4 1 2 3 4	The set is accepted.

Necklace

Task code: NECKLACE Time limit: 1 sec

There are N points marked on a surface, pair ($\mathbf{x_i}$, $\mathbf{y_i}$) is coordinates of a point number **i**. Let's call a **necklace** a set of **N** figures which fulfills the following rules.

- The figure #**i** consists of all such points (**x**, **y**) that $(\mathbf{x} - \mathbf{x_i})^2 + (\mathbf{y} - \mathbf{y_i})^2 \le \mathbf{r_i}^2$, where $\mathbf{r_i} \ge 0$.
- Figures #**i** and #(**i+1**) intersect ($1 \le \mathbf{i} < \mathbf{N}$)
- Figures #1 and #**N** intersect
- All the rest pairs of figures do not intersect.

Write a program which takes points and constructs a necklace.

The input data are in the text file NECKLACE.IN. The first line of the file contains one integer number **N** ($2 \le \mathbf{N} \le 100$). Each of the next **N** lines contains two real numbers $\mathbf{x_i}$, $\mathbf{y_i}$ ($-1\,000 \le \mathbf{x_i}, \mathbf{y_i} \le 1\,000$), separated by one space. It is guaranteed that at least one necklace exists.

The output text file NECKLACE.OUT has to contain **N** lines. A line #**i** contains real number $\mathbf{r_i}$, ($0 \le \mathbf{r_i} < 10\,000$). To avoid potential accuracy problems, a checking program uses the following rules.

- Figures #**i** and #**j** definitely do not intersect if $(r_i + \mathbf{r_j}) \le \mathbf{d_{ij}} + 10^{-5}$
- Figures #**i** and #**j** definitely intersect if $(\mathbf{d_{ij}} - 10^{-5}) \le (\mathbf{r_i} + \mathbf{r_j})$.
- The case when $\mathbf{d_{ij}} - 10^{-5} < (\mathbf{r_i} + \mathbf{r_j}) < \mathbf{d_{ij}} + 10^{-5}$ is decided in favour of a contestant.
- d_{ij} equals $\mathbf{sqrt((x_i – x_j)^2 + (y_i – y_j)^2)}$ in the rules above.

Example of input data	Example of output data

4
0 0
10 0
10 10
0 10
	7
	7
	7
	7

Divisibility by 15

Task code: DIV15 Time limit: 1 sec

There is a string containing only decimal digit characters. The length of the string is between 1 and 1000. Using characters of the string, you have to construct the maximum number which divides by fifteen without remainder. Each character of the string may not be used more than once.

The input file DIV15.IN contains a single line representing the source string.

The output file DIV15.OUT contains one line with the decimal representation of the maximum number. Leading zeroes should be omitted. If no number can be constructed, the output file has to contain a single word "**`impossible`**".

Example of input data	Example of output data
`02041`	`4200`

The lazy programmer

Task code: LAZY Time limit: 1 sec

A new web-design studio, called *SMART* (Simply Masters of ART), employs two people. The first one is a web-designer and an executive director at the same time. The second one is a programmer. The director is so a nimble guy that the studio has already got **N** contracts for web site development. Each contract has a deadline $\mathbf{d_i}$.

It is known that the programmer is lazy. Usually he does not work as fast as he could. Therefore, under normal conditions the programmer needs $\mathbf{b_i}$ of time to perform the contract number **i .** Fortunately, the guy is very greedy for money. If the director pays him $\mathbf{x_i}$ dollars extra, he needs only $\mathbf{(b_i - a_i\, x_i)}$ of time to do his job. But this extra payment does not influent other contract. It means that each contract should be paid separately to be done faster. The programmer is so greedy that he can do his job almost instantly if the extra payment is $\mathbf{(b_i/a_i)}$ dollars for the contract number **i** .

The director has a difficult problem to solve. He needs to organize programmer's job and, may be, assign extra payments for some of the contracts so that all contracts are performed in time. Obviously he wishes to minimize the sum of extra payments. Help the director!

The input data are in the text file LAZY.IN. The first line contains the number of contracts **N** ($1 \le \mathbf{N} \le 100\,000$, integer). Each of the next **N** lines describes one contract and contains integer numbers $\mathbf{a_i}$, $\mathbf{b_i}$, $\mathbf{d_i}$ ($1 \le \mathbf{a_i}, \mathbf{b_i} \le 10\,000$; $1 \le \mathbf{d_i} \le 1\,000\,000\,000$) separated by spaces.

The output text file LAZY.OUT needs to contain a single real number **S** in the only line of file. **S** is the minimum sum of money which the director needs to pay extra so that the programmer could perform all contracts in time. The number must have two digits after the decimal point.

Example of input data

```
2
20 50 100
10 100 50
```

Example of output data

```
5.00
```

“Switchback”

Task code: BUILDER Time limit: 3 sec

A construction firm, which you work for, has signed a contract to complete construction of attraction called "Switchback". The building site is a rectangle which is split into **N**x**M** squares ($1 \le$ **N**, **M** ≤ 50), **N** rows down, **M** columns across. The square in the north-west corner of the building site is (1, 1), while the square in the south-east corner is (**N**, **M**).

By the moment of signing the contract the following work has already been done at the attraction building site. First, a mast was built in each of the squares. The height of the mast in the square (**i**, **j**) is $\mathbf{H_{ij}}$ ($1 \le \mathbf{H_{ij}} \le 100$, natural). Second, construction of a launching complex was started at the top of the mast with coordinates (**a**, **b**), where $1 \le$ **a** $\le$ **N**, $1 \le$ **b** $\le$ **M**.

The idea of the attraction is the following. A small carriage with passengers moves from one square to another adjacent square, square by square. It starts moving from the square with the launching complex and moves until it stops. It moves on rails built on tops of masts according to the following rules:

- ❖ at the launching complex the carriage is slightly pushed so to start moving in any direction **downwards**;
- ❖ on arrival to the next square the carriage can continue its movement in the same direction or can turn 90° to the right or to the left;
- ❖ the carriage’s speed increases by one for each unit of height reduction while moving downwards; the carriage **can not** move downwards if its current speed is zero;
- ❖ the carriage’s speed decreases by two for each unit of height increment while moving upwards. If the carriage moves upwards than its initial speed has to be sufficient to reach the next square;
- ❖ the carriage’s speed during a transition to an adjacent square decreases by one while moving horizontally;
- ❖ safety rules forbid to change the carriage’s speed during transition to an adjacent square by more than 2 units;
- ❖ the carriage has brakes which can be used only to stop the carriage on arrival to the final square. However, safety rules forbid using brakes if the carriage’s speed on arrival to this square is more than three units.

The number of squares visited by the carriage during its trip is called **attraction length**. If the carriage visits a square more than once than each of the visits counts. The first square of the path does not count.

The construction firm has to build rails on tops of the masts. As the only programmer in the firm, you need to write a program which can find the longest possible path according to the rules above. It is guaranteed that this path exists and its length is greater than zero.

The input data are in the text file BUILDER.IN. The first line contains values **N, M, a, b**. The next N lines contain values $\mathbf{H_{i,j}}$, **M** values per line. The numbers in the input file’s lines are separated by one or more spaces.

The output file BUILDER.OUT contains several lines. The first line contains an attraction length **k**. Next k lines contain coordinates of squares and describe the longest path.

Example of input data

```
5 5 1 1
10 8 6 3 4
9  7 6 5 8
4  5 5 2 1
2  3 5 7 8
1  4 3 3 2
```

Example of output data

```
14
2 1
2 2
1 2
1 3
2 3
3 3
3 2
3 1
4 1
4 2
5 2
5 3
5 4
5 5
```

Laser-Ball

Task code: LASER Time limit: 3 sec

Mr. X and his friends are great funs of Laser-Ball Game. Basically, the main idea of the game is very simple. Each player of the game has a laser gun and it is necessary to hit an opponent with a laser beam. To make the game more interesting, Mr. X with friends created their own Laser-Ball playground. They rented an empty rectangular hall and set up a number of big mirrors. What makes the game more interesting is the additional ability to shoot into a mirror since that a reflected laser beam also can hit an opponent.

You need to write a program which determines if it is possible to hit an opponent standing in a point B from point A. If such a hit is possible the program needs to give a direction.

To clarify the problem the following rules are given.

- There are **N** mirrors in the hall.
- Each mirror is a rectangle. It is standing vertically on one of its edges. Therefore, a mirror can be described by a pair of coordinates $(X_1, Y_1) - (X_2, Y_2)$, where $(X_1, Y_1) \neq (X_2, Y_2)$ and X_i, Y_i are real numbers. The height of a mirror is not essential for this problem.
- Mirrors neither touch nor cross each other.
- Both sides of a mirror have a reflection layer. A mirror reflects a laser beam according to physical laws. Let's assume that an edge reflects in the same way as the inner part of a mirror.
- Let's assume that a mirror is absolutely thin. Thus, a laser beam can pass parallel to the mirror as close as necessary, even at 0 distance (see example 2 below)
- A laser beam hits an opponent if it passes the opponent not farther then $\mathbf{10^{-4}}$
- A laser beam goes out of point **A(0, 0)**
- A opponent is in the point $\mathbf{B(X_B, Y_B)}$
- A direction of a shoot, which your program needs to calculate, is a vector in a form **(dx, dy)**
- When a laser beam hits a mirror it loses some part of its energy. After the beam hit mirrors **(K+1)** times it can't hit an opponent. Any mirror can be hit more than once.
- $0 \le \mathbf{N} \le 100$, $1 \le \mathbf{K} \le 10$, but $N^K \le 10^6$

The input file LASER.IN contains one data set. A data set starts with a line containing two real numbers X_B and Y_B separated by one or more spaces. The next line contains two integer numbers, N and K, separated by one or more spaces. The next N lines contain four real numbers $X1_i$, $Y1_i$, $X2_i$, $Y2_i$ each, separated by one or more spaces.

The output file LASER.OUT contains the word "**impossible**", if the shoot is impossible. Otherwise it contains two real numbers **a, b,** (separated by one space), which give a direction of the shoot.

Example #1 of input data

```
4.0 0.0
2 1
2 -1 2 1
1 2 3 2
```

Example #1 of output data

```
1 1
```

Example #2 of input data

```
4 0
1 1
1 0 3 0
```

Example #2 of output data

```
1 0
```

Example #3 of input data

```
4 0
1 1
2 -1 2 1
```

Example #3 of output data

```
impossible
```