2012 ACM ICPC Southeast USA Regional Programming Contest

10 November, 2012

Division I

PROBLEMS

Hosted by:

Florida Institute of Technology

Georgia Institute of Technology

University of West Florida

A: Candy Store

You are walking with a friend, when you pass a candy store. You make a comment about how unhealthy their wares are. Your friend issues an interesting challenge: who can be the unhealthiest? Both of you will go into the store with the same amount of money. Whoever buys candy with the most total calories wins!

Since you're a smart computer scientist, and since you have access to the candy store's inventory, you decide not to take any chances. You will write a program to determine the most calories you can buy. The inventory tells you the price and calories of every item. It also tells you that there is so much in stock that you can buy as much of any kind of candy as you want. You can only buy whole pieces of candy.

Input

There will be multiple test cases in the input. Each test case will begin with a line with an integer ***n*** (1≤***n***≤5,000), and an amount of money ***m*** ($0.01≤***m***≤$100.00), separated by a single space, where ***n*** is the number of different types of candy for sale, and ***m*** is the amount of money you have to spend. The monetary amount ***m*** will be expressed in dollars with exactly two decimal places, and with no leading zeros unless the amount is less than one dollar. There will be no dollar sign. Each of the next ***n*** lines will have an integer ***c*** (1≤***c***≤5,000) and an amount of money ***p*** ($0.01≤***p***≤$100.00), separated by a single space, where ***c*** is the number of calories in a single piece of candy, and ***p*** is the price of a single piece of candy, in dollars and in the same format as ***m***. The input will end with a line containing '**0 0.00**'.

Output

For each test case, output a single integer, indicating the maximum amount of calories you can buy with up to ***m*** dollars. Output no spaces, and do not separate answers with blank lines.

Sample Input	**Sample Output**

`2 8.00`
`700 7.00`
`199 2.00`
`3 8.00`
`700 7.00`
`299 3.00`
`499 5.00`
`0 0.00`
	`796`
	`798`

B: Collision Detection

As a preliminary step in developing an autonomous vehicle system, your team is seeking to prove that a central traffic controller can sound an alert when automobiles are likely to collide unless corrective actions are taken.

The test course consists of a number of straight tracks that intersect at a variety of angles. As cars pass sensors mounted on the tracks, their position and speed is recorded and sent to the central controller. The controller remembers its two most recent sets of readings for each car.

We want the controller to sound the alert whenever two cars will, if they behave as predicted, pass 'dangerously close' to one another any time within the next 30 seconds (following the most recent of the sensor readings). For this purpose, consider that cars are dangerously close if they pass within 18ft. of one another. Cars are considered safe if their closest approach is at least 20ft apart. A passage within 18ft to 20ft is considered ambiguous.

Assume that

- The cars remain on their straight course
- The acceleration (change in speed per unit time) of each car remains constant over the time between observations and for the next 30 sec, with the two exceptions given below.
    - Exception 1: if the car is decelerating, it stops decelerating if its speed reaches zero (cars do not shift into reverse)
    - Exception 2: if the car is accelerating, it stops accelerating if its speed reaches 80 feet per second (about 55 MPH).

Given the two most recent sets of reading for each of two cars, determine if they will pass within 18ft of each other within 30 seconds of the last measurement.

Input

There will be multiple test cases in the input. Each test case consists of four observations, one observation per line. The first two observations are for the first car, the second two observations are for the second car.

Each observation consists of four floating point numbers ***t***, ***x***, ***y*** and ***s***, separated by single spaces, where:

- ***t*** is the time of the observation in seconds (0≤***t***≤120)
- ***x*** and ***y*** give the position of the car at the time of the observation, in feet (-5,000≤***x***,***y***≤5,000)
- ***s*** is the speed in feet per second (0≤***s***≤80)

There will be no data sets in which the closest approach within the indicated timer interval falls in the ambiguous 18ft to 20ft. range. The two observations for a given car will always occur at distinct times, and the first time for each car will be before the second time for that car.

Input is terminated with a line with four negative numbers.

Output

For each data set, print a single line consisting of either **1** if the cars will come within 18 feet of each other within 30 seconds following the maximum of the 4 input times, or **0** if not. Output no extra spaces, and do not separate answers with blank lines.

Sample Input	Sample Output

`10 0 0 10`
`11 7.42 7.42 11`
`11 41.0 106.0 16`
`12 56  106 14`
`0 0 0 50`
`0.5 21.7 12.5 50.1`
`0.25 39.0 22.5 50`
`0.75 60.7 35.0 50.1`
`-1 -1 -1 -1`
	`1`
	`0`

C: Component Testing

The engineers at ACM Corp. have just developed some new components. They plan to spend the next two months thoroughly reviewing and testing these new components. The components are categorized into several different classes, depending on their complexity and importance. Components in different classes may require different number of reviewers, whereas components in the same class always require the same number of reviewers.

There are also several different job titles at ACM Corp. Each engineer holds a single job title. All engineers holding a given job title have the same limit on the number of components that they can review. Note that an engineer can be assigned to review any collection of components and will be able to complete the task, regardless of which classes the components belong to. An engineer may review some components of the same class, and others from different classes, but an engineer cannot review the same component more than once.

Can the engineers complete their goal and finish testing all components in two months?

Input

There will be multiple test cases in the input. The first line of each test case contains two integers ***n*** (1≤***n***≤10,000) and ***m*** (1≤***m***≤10,000), where ***n*** is the number of component classes and ***m*** is the number of job titles. The next ***n*** lines each contains two integers ***j*** (1≤***j***≤100,000) and ***c*** (0≤***c***≤100,000), representing that there are ***j*** components in this class and that each of them requires at least ***c*** different reviewers. Then, the next ***m*** lines each contains two integers ***k*** (1≤***k***≤100,000) and ***d*** (0≤***d***≤100,000), representing that there are ***k*** engineers with this job title and that each of them may be assigned to review at most ***d*** components. The input will end with a line with two **0**s.

Output

For each test case, output **1** if it is possible for the engineers to finish testing all of the components, and **0** otherwise. Output no spaces, and do not separate answers with blank lines.

Sample Input	Sample Output

```
3 2
2 3
1 2
2 1
2 2
2 3
5 2
1 1
1 3
1 1
1 3
1 1
1 20
1 4
0 0
```
	```
	1
	0
	```

D: Do It Wrong, Get It Right

In elementary school, students learn to subtract fractions by first getting a common denominator and then subtracting the numerators. However, sometimes a student will work the problem incorrectly and still arrive at the correct answer. For example, for the problem

$$\frac{5}{4} - \frac{9}{12}$$

one can subtract the numbers in the numerator and then subtract the numbers in the denominator, simplify and get the answer. i.e.

$$\frac{5}{4} - \frac{9}{12} = \frac{-4}{-8} = \frac{4}{8} = \frac{1}{2}$$

For a given fraction ***b***/***n***, your task is to find all of the values ***a*** and ***m***, where ***a***≥**0** and ***m***>**0**, for which

$$\frac{a}{m} - \frac{b}{n} = \frac{a-b}{m-n}$$

Input

There will be several test cases in the input. Each test case will consist of a single line with two integers, ***b*** and ***n*** ($1 \le b, n \le 10^6$) separated by a single space. The input will end with a line with two **0**s.

Output

For each case, output all of the requested fractions on a single line, sorted from smallest to largest. For equivalent fractions, print the one with the smaller numerator first. Output each fraction in the form "***a/m***" with no spaces immediately before or after the "**/**". Output a single space between fractions. Output no extra spaces, and do not separate answers with blank lines.

Sample Input	**Sample Output**
`9 12`	`0/24 5/20 8/16 8/8 5/4`
`12 14`	`0/28 9/21 9/7`
`4 12`	`0/24 3/18 3/6`
`0 0`	

E: Funhouse

An amusement park is building a new walk-through funhouse. It is being built in a large space: 1000ft x 1000ft. The park will build walls in the space, separating it into rooms. Some walls will have doors so that guests can move between rooms. Guests will enter through specially marked entrances, and exit through specially marked exits. They can move through the space as they wish – in fact, there may be many different ways of moving from the entrances to the exits. Of course, there will be many amusing things along the way.

The park wants to install "shakerboards", which are moving floors, to surprise the guests. To make them less obvious, wherever they're installed, they'll fill the whole room. That way, they won't stand out. The designers want every guest in the funhouse to experience shakerboards – but, as you can imagine, shakerboards are expensive, so the park wants to cover as little space with them as possible.

Given a description of a funhouse design, what's the smallest area that must be covered with shakerboards to assure that every guest experiences them?

Input

There will be several data sets. Each data set will begin with a line with one integer ***n*** (3≤***n***≤1,000), which is the number of walls. Each of the next ***n*** lines will describe a wall, like this:

x1 y1 x2 y2 EXDW

where (***x1***,***y1***) and (***x2***,***y2***) are the endpoints of the wall, and ***EXDW*** is a single capital letter: '`E`' for an Entrance, '`X`' for an Exit, '`D`' for an interior wall with a door, and '`W`' for any wall without a door. '`E`' and '`X`' are guaranteed to only appear on exterior walls, and '`D`' is guaranteed to only appear on interior walls. '`W`' may appear on either. The endpoint coordinates will be integers, with values between 0 and 1,000 inclusive. Walls will never touch each other in any way, except for sharing endpoints. Every endpoint will be coincident with another wall's endpoint. No wall will have zero length. Every entrance is guaranteed to have at least one path to an exit, and every exit is guaranteed to have at least one path from an entrance. The funhouse will consist of a single building. In order to provide power throughout the building, every interior wall is connected to an exterior wall either directly or indirectly via a series of other walls.

End of input will be marked by a line with a single **0**.

Output

For each test case, output a single floating point number indicating the smallest area that the park owners must cover with shakerboards so that every guest in the funhouse will experience them. Output this number with exactly one decimal place. Output no spaces, and do not separate answers with blank lines.

Sample Input	Sample Output
6 0 0 100 0 W 0 0 0 100 E 0 100 100 100 W 100 0 100 100 D 100 0 200 0 W 200 0 100 100 X 14 0 0 100 0 W 100 0 110 0 E 110 0 190 0 W 190 0 200 0 E 0 0 0 100 W 100 0 100 100 D 200 0 200 100 W 0 100 100 100 D 100 100 200 100 D 0 100 0 150 X 100 100 100 150 D 200 100 200 150 X 0 150 100 150 W 100 150 200 150 W 0	5000.0 10000.0

F: A Terribly Grimm Problem

Grimm's conjecture states that to each element of a set of consecutive composite numbers one can assign a distinct prime that divides it.

For example, for the range 242 to 250, one can assign distinct primes as follows:

242	243	244	245	246	247	248	249	250
2	3	61	7	41	13	31	83	5

Given the lower and upper bounds of a sequence of composite numbers, find a distinct prime for each. If there is more than one such assignment, output the one with the smallest first prime. If there is still more than one, output the one with the smallest second prime, and so on.

Input

There will be several test cases in the input. Each test case will consist of a single line with two integers, ***lo*** and ***hi*** ($4 \le lo < hi \le 10^{10}$), separated by a single space. It is guaranteed that all the numbers in the range from ***lo*** to ***hi*** inclusive are composite. The input will end with a line with two **0**s.

Output

For each test case, output the set of unique primes, in order, all on the same line, separated by single spaces. Output no extra spaces, and do not separate answers with blank lines.

Sample Input

```
242 250
8 10
0 0
```

Sample Output

```
2 3 61 7 41 13 31 83 5
2 3 5
```

G: Heads or Tails

I am the proud owner of a beautiful, classic, electronic puzzle. It consists of a matrix with ***n*** rows and ***m*** columns filled with coins. Whenever I select a coin, it will change from heads to tails or vice versa. Moreover, all the coins in the cells of the puzzle that share a side with it will also change.

I also have a pesky little sister. She likes to play with my beautiful puzzle. One day, she said to me "I've been playing with your puzzle! I pushed ***k*** coins at random, and now, there are between ***a*** and ***b*** heads showing!" She wouldn't tell me much more, but I did find out that she didn't remember whether she pressed any coin more than once. In other words, she truly struck at random; she didn't take care to press ***k*** different coins.

This got me to thinking: What must the board have looked like before she started playing with it? Of course, there are many starting board positions which would have an expected number of heads between ***a*** and ***b*** after ***k*** random moves. If I were to put them all in order (alphabetically, starting with the first row, and then the second, and so on, using '**H**' for a Head and '**T**' for a Tail), which one would be the ***i***th one in the list?

Input

There will be several data sets. Each data set will consist of a single line with six integers:

n m k a b i

where ***n*** and ***m*** ($1 \le n,m \le 7$) are the dimensions of the board, ***k*** ($0 \le k \le 50$) is the number of random moves my little sister made, ***a*** and ***b*** ($0 \le a \le b \le n*m$) represent the range of the number of Heads in the ending state of the board, and ***i*** ($1 \le i \le 2^{m*n}$) is the position in a sorted list of starting board positions that I'm interested in. The input will end with a line with six **0**s.

Output

For each test case, output ***n*** rows of ***m*** characters, representing the requested starting board position. Use '**H**' for a Head, and '**T**' for a Tail. Your output should have no other printable characters, or spaces. Do not separate answers with blank lines.

Sample Input

```
1 5 7 2 3 10
2 2 1 1 2 6
3 4 11 3 12 3668
0 0 0 0 0 0
```

Sample Output

```
HTHHT
HT
HT
TTTH
HTHT
HHTT
```

H: Tsunami

The country of Cartesia can be described simply by a Cartesian plane. The x-axis is a shoreline. The positive y half-plane is land, and the negative y half-plane is ocean. Several large cities dot the mainland. Their positions can be described by coordinates (x,y), with y>0. Unfortunately, there are sometimes tsunamis in the ocean near Cartesia. When this happens, the entire country can flood. The waters will start at y=0 and advance uniformly in the positive y direction.

Cartesia is trying to develop a tsunami warning system. The warning system consists of two components: a single meteorological center which can detect a tsunami miles out, and wired connections which can carry the warning from city to city in straight lines (No wireless communication!!). Despite the lack of wireless communications, the wired connections can carry the warning virtually instantaneously.

A city is considered *safe* if it either has the meteorological center, or if it has a direct wire connection to another *safe* city (i.e. if it has a multi-hop cable path to the single meteorological center).

Unfortunately, a simple engineering problem is made more complicated by politics! If a city A receives the warning from city B, and city B is further away from the shore than city A, then city A's leaders will complain! "We're closer to the ocean than city B, so the warning should have gone through us!" With a sigh, you agree to find a solution where no city will get the warning from a city that's further from the shore. (This means that the meteorological center must be in a city that's closest to the shore.)

Given a description of Cartesia, find the least amount of cable necessary to build a tsunami warning system where every city is *safe*, and no city will receive the warning from another city that is further from the shore.

Input

There will be several test cases in the input. Each test case will begin an integer ***n*** (1≤***n***≤1,000) on its own line, indicating the number of cities. On each of the next ***n*** lines will be a pair of integers ***x*** and ***y*** (-1,000≤***x***≤1,000, 0<***y***≤1,000), each of which is the (***x***,***y***) location of a city. These integers will be separated by a single space. Within a test case, all (***x***,***y***) pairs will be unique. The input will end with a line containing a single **0**.

Output

For each test case, output a single real number on its own line, which is the minimum amount of cable which must be used to build the tsunami warning system. Output this number with exactly two decimal places. Output no extra spaces, and do not separate answers with blank lines.

Sample Input	Sample Output

```
3
100 10
300 10
200 110
4
100 10
300 10
200 110
200 60
0
```
	```
	341.42
	361.80
	```

I: Unhappy Numbers

Numbers have feelings too! For any positive integer, take the sum of the squares of each of its digits, and add them together. Take the result, and do it again. A number is Happy if, after repeating this process a finite number of times, the sum is 1. Some happy numbers take more iterations of this process to get to 1 than others, and that would be referred to as its distance from happiness. 1's distance from happiness is 0. 23's distance from happiness is 3, since $2^2 + 3^2 = 13$, $1^2 + 3^2 = 10$, and $1^2 + 0^2 = 1$. Numbers are Unhappy if they are infinitely far away from happiness because they get stuck in a loop.

Given the lower end and upper end of a range of integers, determine how many Unhappy numbers are in that range (inclusive).

Input

There will be several test cases in the input. Each test case will consist of two positive integers, ***lo*** and ***hi*** ($0<lo\leq hi\leq 10^{18}$) on a single line, with a single space between them. Input will terminate with two **0**s.

Output

For each test case, output a single integer on its own line, indicating the count of Unhappy Numbers between ***lo*** and ***hi*** (inclusive). Output no extra spaces, and do not separate answers with blank lines.

Sample Input	Sample Output
1 10	7
1 100	80
0 0	

J: Walls

There are a number of research stations on a featureless patch of desert, which can be modeled as a Cartesian plane. Each station is located at some point (***x***,***y***) where ***x*** and ***y*** are even integers. For security reasons, sufficiently long and high walls are to be constructed to separate the stations so that no station is visible from any of the other stations. A wall may only be constructed along a North-South or East-West line. A vertical wall may be built at an odd x-coordinate, and a horizontal wall may be built at an odd y-coordinate. Since the stations are located at even valued coordinates, and the walls are built along odd valued coordinates, no wall can ever touch a station. The walls are always long enough to completely separate stations on one side from stations on the other side.

Given a list of stations, you must determine the smallest number of walls that need to be constructed.

Input

There will be several test cases in the input. Each test case will begin with an integer ***n*** (2≤***n***≤100), which is the number of stations. The next ***n*** lines each will contain two integers ***x*** and ***y*** (0≤***x***,***y***≤36), separated by a single space, indicating the (***x***,***y***) location of a station. The ***x*** and ***y*** values are guaranteed to be even. Within a test case, all (***x***,***y***) locations will be unique. The last test case will be followed by a line with a single **0**.

Output

For each test case, output a single integer, indicating the smallest number of walls that can prevent the given ***n*** stations from seeing each other. That is, a straight line-segment joining any two stations must be intersected by at least one wall. Output no extra spaces, and do not separate answers with blank lines.

Sample Input	Sample Output

```
4
12 12
4 8
8 6
2 4
4
0 0
4 4
10 8
14 6
0
```
	```
	2
	3
	```