UW-Madison's

2009 ACM-ICPC Individual Placement Test

October 4th, 1:00-6:00pm, CS1350

Overview:
This test consists of seven problems, which will be referred to by the following names (respective of order):
numbers, willy, island, ferry, fence, rockers, corners.

Input/Output:
Your programs should take input from standard in (i.e., the keyboard) and output to standard out (i.e., the terminal). As is standard practice for the ICPC, you may assume that the input strictly follows the description in the problems. It is your responsibility to ensure that your output precisely matches that described in the problems, or else risk your program being rejected with a “Presentation Error”.

Problem Submission:
Problem submission will be handled via the PC^2 judging software, as at the actual competition.

Clarifications:
Clarification requests will also be handled via the PC^2 software. Accepted clarification requests will be answered to all those taking the test. Experience of previous years learns that most clarification requests are rejected, and receive a simple response such as “Read the problem description”.

Printing:
You may print to the printer at any time during the test.

Written Solutions:
We encourage you to spend the last half hour of the test to write down the main idea behind the solution to any of the problems for which you have not had a program accepted. Please be concise, using at most a few sentences within the space provided on the opposite side of this page. We will take these partial solutions into account along with your official ranking when composing teams, although they will have less weight.

After the Test:
The proctor will announce when time is up. Please stop working at this time and take a moment to fill out the form on the back of this sheet and turn it in to the proctor (you may keep the problems). You are invited to join us for pizza and soda after the test in 1325 CS.

Information:

Name: _____

CS Login:_____

PC^2 Login/Pass: _____ / _____

Student Status (e.g Junior, 1st-year grad):_____

Year of birth:_____, Year starting college:_____

How many ICPC regionals have you participated in?____ How many world finals?___

Which of C/C++/Java do you prefer:____ Please indicate your proficiency in each:

Which classes have you taken (or are taking) whichare relevent to the ICPC?

What do you feel your strengths are with respect to the ICPC?

Are you able to travel to the world finals in Harbin, China, February 1 - 6, 2010 (and obtain a visa / passport as necessary)?

If your team progresses to the world finals, how many hours per week could you commit to practicing, starting mid-November? ____

Are there people you would prefer to be (or not be) on the same team as?

Is there anything else we should know about you?

Written solutions for unsolved problems:

numbers:

willy:

island:

ferry:

fence:

rockers:

corners:

2687 – Amusing Numbers

Europe – Northeastern – 2002/2003

Let us consider the set of integer numbers between 1 and N inclusive. Let us order them lexicographically (i. e. like in the vocabulary), for example, for $N = 11$ the order would be: 1, 10, 11, 2, 3, 4, 5, 6, 7, 8, 9.

Let us denote the position of the number K in this ordering as $Q_{N, K}$. For example, $Q_{11, 2} = 4$. Given numbers K and M find the smallest N such that $Q_{N, K} = M$.

Input

Input file contains several test cases, one per line. Each of them consists of two integer numbers K and M ($1 \leq K, M \leq 10^9$) separated by a space.

Output

For each input case write a different output line. If such N that $Q_{N, K} = M$ exists then write the smallest such N, otherwise write `0'.

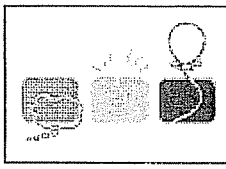
Sample Input

```
2 4
2 1
100000001 1000000000
1000000000 11
```

Sample Output

```
11
0
100000000888888879
0
```

Northeastern 2002–2003



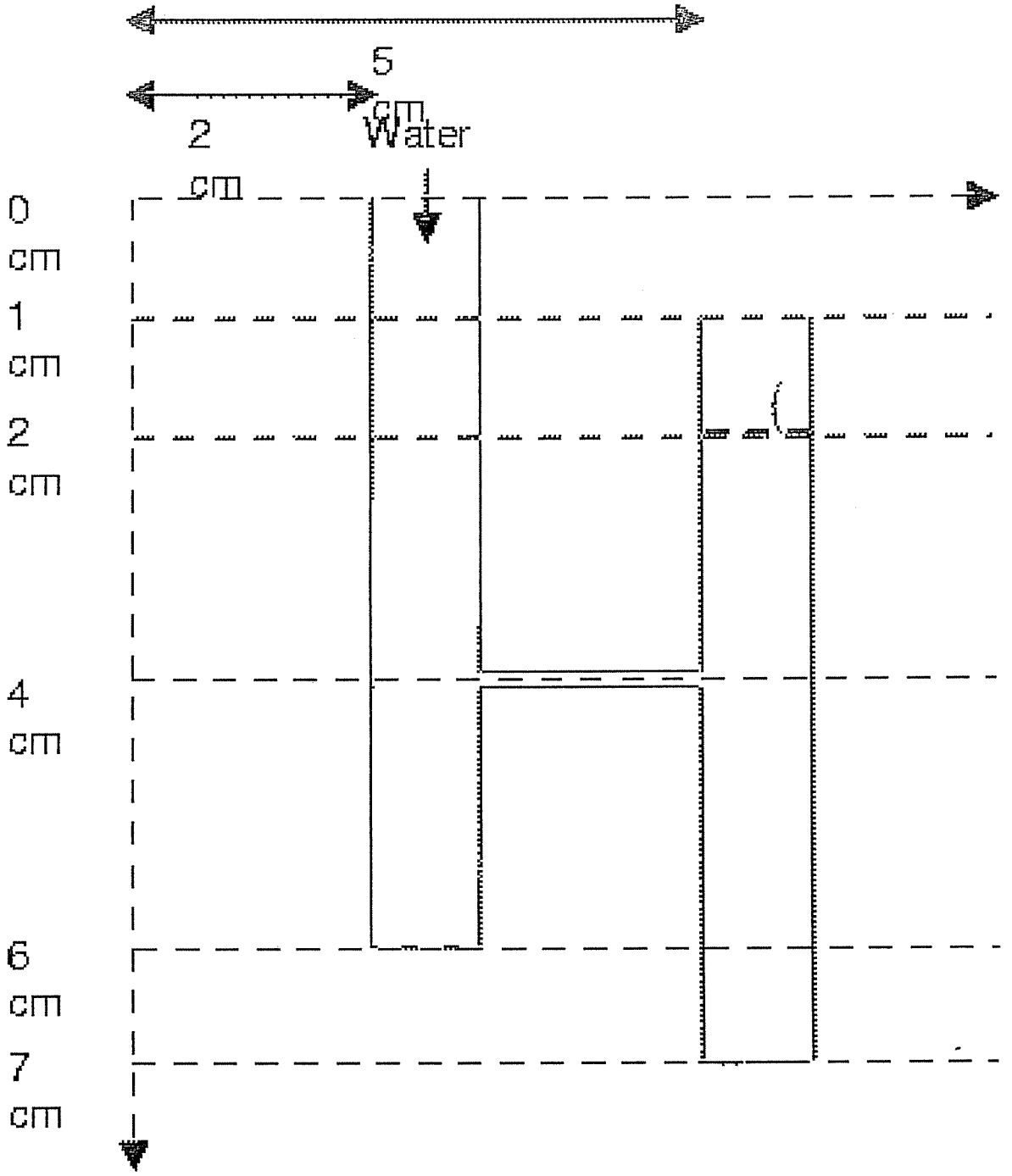
2343 – The Willy Memorial Program

Asia – Tehran – 2001/2002

Willy the spider used to live in the chemistry laboratory of Dr. Petro. He used to wander about the lab pipes and sometimes inside empty ones. One night while he was in a pipe, he fell asleep. The next morning, Dr. Petro came to the lab. He didn't notice Willy while opening the valve to fill the pipes with hot water. Meanwhile, Stanley the gray mouse got what was going to happen. No time to lose! Stan ran hard to reach the valve before Willy gets drawn, but... Alas! He couldn't make it!

Poor Willy was boiled in hot water, but his memory is still in our hearts. Though Stan tried his best, we want to write a program, in the memory of Willy, to compute the time Stan had, to rescue Willy, assuming he started to run just when the doctor opened the valve.

To simplify the problem, assume the pipes are all vertical cylinders with diameter 1 cm. Every pipe is open from the top and closed at the bottom. Some of the pipes are connected through special horizontal pipes named *links*. The links have very high flow capacity, but are so tiny that at any given time, the volume of water inside them is negligible. The water enters from top of one of the pipes with a constant rate of 0.25 π cm^3/sec and begins to fill the pipe from the bottom until the water reaches a link through which it flows horizontally and begins to fill the connected pipe. From elementary physics we know if two pipes are connected and the surface of the water is above the connecting link, the level of water in both pipes remains the same when we try to fill one of them. In this case the water fills each pipe with a rate equal to half of the rate of incoming water. As an example, consider the following configuration:



First, the lower 2 centimeters of the left pipe is filled with water at full rate, then, the lower 3 centimeters of the right pipe is filled, and after that, the upper part of the two pipes are filled in parallel at half rate. The input to your program is a configuration of pipes and links, and a target level in one of the pipes (the heavy dotted line in the above figure). The program should report how long it takes for the level of water to reach the target level. For the above configuration, the output is 9 seconds.

It is assumed that the water falls very rapidly, such that the time required for the water to fall can be neglected. The target level is always assumed to be a bit higher than the specified level for it. As an example, if we set the target point to level 4 in the left pipe in the figure above, the elapsed time for water to reach that target is assumed to be 5 (not 2), Also note that if the water reaches to the top of a pipe (say in level x), it won't pour out outside the pipe until empty spaces in connected pipes below level x are filled (if can be filled, i.e. the level of water reaches the connecting links). (Note that there may be some links at level x, to which water is entered). After all such spaces are filled; the water level would not go up further.

Input

To describe positions, we assume the coordinates are expressed as (x, y) and the origin lies in the top–left of all pipes and links. (Note that y coordinates are increased downwards). All coordinates are integer numbers between 0 and 100, inclusive.

The first line of the input file contains a single integer t ($1 \leq t \leq 10$), the number of test cases, followed by the input data for each test case. The first line of each test case is p ($1 \leq p \leq 20$), the number of pipes, followed by p lines, each describing a pipe. Each pipe description line consists of three numbers. The first two are (x, y) coordinates of the upper–left corner of the pipe and the third number is the height of the pipe (at least 1 cm and at most 20 cm). Note that diameter of each pipe is 1 cm.

After input data describing the pipes, there is a line containing a single integer l, which is the number of links ($0 \leq l \leq 50$). After it, there are l lines describing links. Each link description contains 3 integers. The first two are (x, y) coordinates of the left end–point of the link and the third is the length of the link (at least 1 cm and at most 20 cm). It is assumed that the width of the link is zero.

The last line for each test case contains two numbers. The first is the number of target pipe (starting from one, with the order appeared in test data). The second line is the desired y for the level of water in the target pipe (note that the specified level may be out of the pipe at all).

You can assume the following about the input:

- The water enters into the first pipe.
- No link crosses a pipe.
- No two links have the same y coordinates.
- No two pipes have the same upper–left x coordinates.
- Both endpoints of each link are connected to pipes.

Output

The output file should contain exactly t lines with no blank lines in between,each corresponding to one test case. Each output line should contain the time required for the water to reach the target level in the target pipe (an integer number). If in a specific test case, the water never reaches the target level, the line should contain `No Solution' string in it.

Sample Input

```
1
2
2 0 6
5 1 6
1
3 4 2
2 2
```

Sample Output

```
9
```

Tehran 2001–2002

The 2002 26th Annual **acm** International Collegiate

Programming Contest World Finals

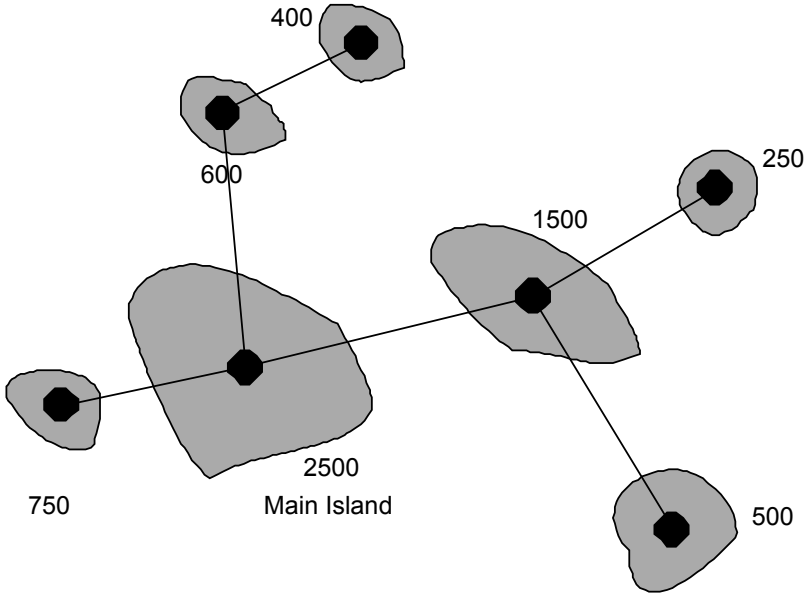
sponsored by IBM

Problem E

Island Hopping

Input: islands.in

The company Pacific Island Net (PIN) has identified several small island groups in the Pacific that do not have a fast internet connection. PIN plans to tap this potential market by offering internet service to the island inhabitants. Each groups of islands already has a deep-sea cable that connects the main island to the closest internet hub on the mainland (be it America, Australia or Asia). All that remains to be done is to connect the islands in a group to each other. You must write a program to help them determine a connection procedure.



For each island, you are given the position of its router and the number of island inhabitants. In the figure, the dark dots are the routers and the numbers are the numbers of inhabitants. PIN will build connections between pairs of routers such that every router has a path to the main island. PIN has decided to build the network such that the total amount of cable used is minimal. Under this restriction, there may be several optimal networks. However, it does not matter to PIN which of the optimal networks is built.

PIN is interested in the average time required for new customers to access the internet, based on the assumption that construction on all cable links in the network begins at the same time. Cable links can be constructed at a rate of one kilometer of cable per day. As a result, shorter cable links are completed before the longer links. An island will have internet access as soon as there is a path from the island to the main island along completed cable links. If m_i is the number of inhabitants of the i^{th} island and t_i is the time when the island is connected to the internet, then the average connection time is:

$$\frac{\sum t_i * m_i}{\sum m_i}$$

Input

The input consists of several descriptions of groups of islands. The first line of each description contains a single positive integer n, the number of islands in the group ($n \leq 50$). Each of the next n lines has three integers x_i, y_i, m_i, giving the position of the router (x_i, y_i) and number of inhabitants m_i ($m_i > 0$) of the islands. Coordinates are measured in kilometers. The first island in this sequence is the main island.

The input is terminated by the number zero on a line by itself.

Output

For each group of islands in the input, output the sequence number of the group and the average number of days until the inhabitants are connected to the internet. The number of days should have two digits to the right of the decimal point. Use the output format in the sample given below.

Place a blank line after the output of each test case.

Sample Input	Output for the Sample Input
7 11 12 2500 14 17 1500 9 9 750 7 15 600 19 16 500 8 18 400 15 21 250 0	Island Group: 1 Average 3.20

Ferry Loading

Before bridges were common, ferries were used to transport cars across rivers. River ferries, unlike their larger cousins, run on a guide line and are powered by the river's current. Cars drive onto the ferry from one end, the ferry crosses the river, and the cars exit from the other end of the ferry.

There is a ferry across the river that can take *n* cars across the river in *t* minutes and return in *t* minutes. *m* cars arrive at the ferry terminal by a given schedule. What is the earliest time that all the cars can be transported across the river? What is the minimum number of trips that the operator must make to deliver all cars by that time?

The first line of input contains *c*, the number of test cases. Each test case begins with *n, t, m*. *m* lines follow, each giving the arrival time for a car (in minutes since the beginning of the day). The operator can run the ferry whenever he or she wishes, but can take only the cars that have arrived up to that time. For each test case, output a single line with two integers: the time, in minutes since the beginning of the day, when the last car is delivered to the other side of the river, and the minimum number of trips made by the ferry to carry the cars within that time.

You may assume that 0 < n, t, m < 1440. The arrival times for each test case are in non-decreasing order.

Sample input

```
2
2 10 10
0
10
20
30
40
50
60
70
80
90
2 10 3
10
30
40
```

Output for sample input

```
100 5
50 2
```

Problem 4: The Fence Builder

A fence builder has been given a strange task. Provided with *N* (between 3 and 100) pieces of straight fencing, each having an arbitrary length, the builder is to enclose as large a region as possible. The customer wants to know the area of the region that can be enclosed by the fence before it is built. There is only one constraint on the construction: each piece of fencing is connected only at its endpoints to exactly two other different pieces of fencing. That is, after completion, the fence will look like a (possibly irregular) polygon with *N* sides. The customer has guaranteed the builder that the fencing provided will allow for a region with a non-zero area to be enclosed.

Input

There will be multiple cases in the input. For each case, the input begins with the number of pieces of fencing (an integer, *N*). There then follow *N* positive, non-zero real numbers giving the lengths of the fence pieces. A single integer zero follows the last case in the input.

Output

For each case, display the case number (starting with 1) and the maximum area that can be enclosed by the provided fencing materials. Show three fractional digits in each answer. Use the format shown below in displaying the results.

Sample Input

```
3 2.0 2.0 2.0
4 1.0 1.0 1.0 1.0
4 5.0 5.0 3.0 11.0
0
```

Expected Output

```
Case 1: maximum area = 1.732
Case 2: maximum area = 1.000
Case 3: maximum area = 21.000
```

Raucous Rockers

You just inherited the rights to *n* previously unreleased songs recorded by the popular group Raucous Rockers. You plan to release a set of *m* compact disks with a selection of these songs. Each disk can hold a maximum of *t* minutes of music, and a song can not overlap from one disk to another. Since you are a classical music fan and have no way to judge the artistic merits of these songs, you decide on the following criteria for making the selection:

1. The songs will be recorded on the set of disks in the order of the dates they were written - All songs on disk i must be written before those on disk i+1 for 1 <= i < m, and the songs must appear in order on each of these disks.
2. The total number of songs included will be maximized.

Input

The input consists of several datasets. The first line of the input indicates the number of datasets, then there is a blank line and the datasets separated by a blank line. Each dataset consists of a line containing the values of *1 <= n <= 1000*, *1 <= t <= 1,000,000,000* (yes, they like their power ballads) and *1 <= m <= 1,000* (integer numbers) followed by a line containing a list of the length of *n* songs, $t_1, t_2, ..., t_n$ ordered by the date they were written (Each t_i is between 1 and t minutes long, both inclusive, and $\sum_{i=1}^{n} t_i > m \times t$.)

Output

The output for each dataset consists of one integer indicating the number of songs that, following the above selection criteria will fit on *m* disks. Print a blank line between consecutive datasets.

Sample Input

```
2

10 5 3
3, 5, 1, 2, 3, 5, 4, 1, 1, 5

1 1 1
1
```

Sample Output

```
6

1
```

The Twentieth Annual ACM International Collegiate

Programming Contest Finals

sponsored by

Microsoft®

Problem

Cutting Corners

Input file: corner.in

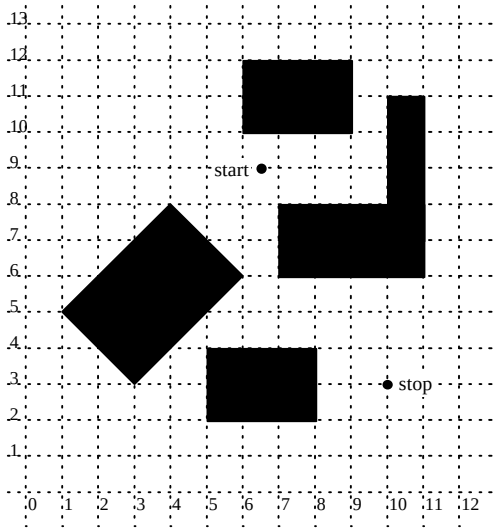
Bicycle messengers delivering documents and small items from one office building to another have long been part of the guerrilla transportation services in several major U.S. cities. The cyclists themselves are a rare breed of riders who are notorious for their speed, their disrespect for one-way streets and traffic signals, and their unflinching bravery in facing motorized vehicles and pedestrians alike.

Bicycle messenger services tend to be very competitive, and Billy's Bicycle Messenger Service is no exception. In order to boost its competitive edge as well as determine its actual expenses, BBMS is developing a new scheme for pricing deliveries that depends in part on the routes messengers travel. You are to write a program to help BBMS determine the minimum distances for various routes.

The following assumptions simplify your task:

- Messengers can ride their bicycles anywhere at ground level except inside buildings.
- Ground floors of buildings consist of rectangles. If two rectangles making up ground floors touch, they share interior space. In that case, they are considered to be part of the same building.
- Two different buildings do not touch, although they can be quite close. (Bicycle messengers— skinny to a fault—can travel between any two different buildings.)
- Starting and stopping points for any given trip are never in the interiors of buildings.
- It is always possible to travel from the starting to stopping point for each trip.

Input for your program will be several scenarios of bicycle delivery trips. Each scenario is a bird's-eye snapshot showing the locations of the buildings and the starting and ending points for a route (all measured with respect to a hypothetical infinite square grid). The picture below is a typical snapshot of buildings, which are shaded, and the route's starting and stopping points. All are superimposed on a grid.



The input file represents several snapshots. Input for each snapshot consists of lines as follows:

First line:	*n*	The number of rectangles comprising buildings in the snapshot (an integer greater than or equal to 0)
Second line:	$x_1\ y_1\ x_2\ y_2$	The *x*- and *y*-coordinates of the starting and stopping points of the route.
Remaining n lines:	$x_1\ y_1\ \ x_2\ y_2\ \ x_3\ y_3$	The *x*- and *y*-coordinates of three vertices of the rectangle representing a rectangular part of a building.

The *x*- and *y*-coordinates of all input data are real numbers between 0 and 1000 inclusive. Successive coordinates on a line are separated by one or more blanks. The end of all input is signified by a "First line" with a negative number of rectangles.

To avoid problems with real precision, the input data set restricts all coordinates to be between 0 and 1000 inclusive. The interior enclosed by any two intersecting rectangles will be at least large enough to contain a square of .01 unit on a side. In addition, two buildings that do not intersect will be at least .01 unit apart.

Output for each snapshot is the number of the input record (snapshot #1, snapshot #2, etc.) and the distance of the shortest path from the starting to stopping points that does not go through the interior of any building. Diasance should be shown with two digits to the right of the decimal. Output for successive snapshots should be separated by blank lines.

The following input data file corresponds to the single snapshot from the illustration on the opposite side.

Sample Input

```
5
6.5   9     10  3
1     5     3   3     6  6
5.25  2     8   2     8  3.5
6     10    6   12    9  12
7     6     11  6     11 8
10    7     11  7     11 11
-1
```

Output for the Sample Input

```
Snapshot #:1
   route distance: 7.28
```