Science Atlantic
Computer Science
Mathematics & Statistics
Conference

2012 ACM-ICPC Atlantic Preliminary Problem Set

Sincere thanks to the following contributors for the problems:
Jalal Almhana, Jim Diamond, Vlado Keselj, Stavros Konstantinidis, Mike McAllister.

Additional great thanks to Dennis MacAlistair Ritchie, whose contributions allow us to be here doing this today, on the first anniversary of his passing.

Hosted at Mount Allison University
October 12, 2012.

Document prepared on Friday 12th October, 2012at 08:41

PROBLEM A: Crazy Trains

Your job is to assemble trains for journeys along your railway. Your job is rather difficult, as your train cars (or coaches) are a hodge-podge, having been purchased at different times from different manufacturers, and therefore have several different styles of hitch in use. The hitches are identified by letter (type A, type B, type C, etc.)

- Each car has two hitches (one at either end), which may be the same, or may be different.
- Type A hitches can only link to other type A hitches, type B only to type B, etc.
- Each engine has one hitch, which must attach to the first car.

Given a particular engine (identified by hitch type) and a set of cars (identified by pairs of hitches), your task is to create a train incorporating as many of the given cars as possible by ordering the cars so that pairs of matching hitches can link the cars together.

Cars can be assigned in any order, and can be linked in either direction (that is, they can be turned around – either the first or second hitch indicated in the input can be used to attach to the car in front, the other hitch must be used to attach to the car that follows).

The first car must link to the hitch on the engine. The last car has an unused hitch at the very end of the train.

Input

The first line of the input indicates how many trains you are asked to create. The first line of each train specification indicates the type of the hitch on the engine. The second line in each train specification indicates how many cars are to be assembled together. The hitch on the engine and on the cars are identified by single capital letters from the Roman alphabet, so there are at most 26 types of hitches (`A`–`Z`).

Output

Your output for each train begins with a line containing four values, each separated by a space from the previous:

1. the word "`train`",
2. the identifier of this train, within the list of trains you are assembling (starting at 1),
3. the maximum number of cars that can be successfully added to the train (which may be as low as zero cars, and as high as the number of cars to be assembled),
4. the type of hitch on the engine

You should then print a list of the cars that are assembled into the longest possible train. Cars should be printed with the "front-most" hitch first, in the order they are finally assembled. The second letter on one line should therefore match the first letter on the next line, as these two hitches are linked in the final train.

You may assume that there will be only one way to assemble the longest train using the cars provided.

Sample input and output are provided on the next page.

Sample input and output for problem A

```
----- Sample Input ---
4
A
1
Z Z
A
7
A B
B B
C B
C D
C E
D E
E F
B
5
B A
B B
C D
C A
B D
A
3
A B
C D
E F
----- Sample Output ---
train 1 0 A
train 2 6 A
A B
B B
B C
C D
D E
E F
train 3 5 B
B A
A C
C D
D B
B B
train 4 1 A
A B
-------end of input/output samples
```

PROBLEM B: Speciation

A group of biologists wandering around in a remote valley in Indonesia has discovered some previously-undocumented mammalian creatures. However, the biologists are unsure as to whether they have discovered just one species or many species, since the individuals look remarkably similar. They have noticed that all of the animals in this small valley spend a certain amount of time each day interacting with each other, and that part of this interaction involves one animal grooming (or being groomed) by another.

After careful observation (and surreptitiously painting a unique ID number on each of the animals, using indelible ink), one of the biologists has formulated the hypothesis that there are, in fact, multiple species, and that each species only grooms other members of its own species.

In order to test this theory out, the biologists have carefully observed the animals over a period of several months. Whenever they see an animal A_1 grooming an animal A_2, they make a note "A_1 -> A_2" in their notebooks. These notes are later transcribed onto a computer for processing.

It is not known whether each animal of a given species grooms every other animal of that species, but the biologists assume that if A_1 grooms A_2 and A_2 grooms A_3, then A_1 is the same species as A_3. They call a group of animals with some grooming connection a grooming group (GG).

The biologists are adept at sitting in trees watching small furry creatures, but not so good at writing programs. Your task is to analyze a set of data and tell the biologists how many different species there are, under the assumption that animals in different GGs are also different species.

The data is broken up into sets of observations lasting one month at a time, but there may be multiple months of observations.

The input consists of:

- a line with the number of months of observations (an integer)
- for each month, there is:
  - an integer N specifying the number of grooming observations
  - N lines of the form "A_1 -> A_2", where A_1 and A_2 are integers between 1 and $99,999$, inclusive.

N is never larger than $99,999$.

For each input month, you are to output (on a line by itself) the number of GGs in that month's data.

```
----- Sample Input ---
1
10
1 -> 2
2 -> 3
3 -> 4
4 -> 2
5 -> 2
6 -> 7
6 -> 8
6 -> 9
6 -> 10
6 -> 11
----- Sample Output ---
2
-------end of input/output samples
```

PROBLEM C: The HBS Language

The members of HBS, the Halifax Binary Society, communicate using binary words that belong to the following strange language (the HBS language): words must be of the form $W_1W_2\ldots W_n$ such that

- n is a positive integer (that is, $n \geq 1$)
- each segment W_i is equal to one of `0`, `01`, `11`

For example, the words `001` and `01111` belong to the HBS language, but the words `111` and `1110` do not. The Society has hired you to write a program that tells whether some given binary words belong to their language.

Your program must read a positive integer N, and then N binary words, one per line. For each input word, the program will output in the next line `YES` or `NO`, depending on whether the word belongs to the HBS language.

```
----- Sample Input 1 ---
6
11001
01111
111
1110
0
100000
----- Sample Output 1 ---
YES
YES
NO
NO
YES
NO
----- Sample Input 2 ---
3
0001
11010
111110
----- Sample Output 2 ---
YES
YES
NO
-------end of input/output samples
```

PROBLEM D: Area of Shapes

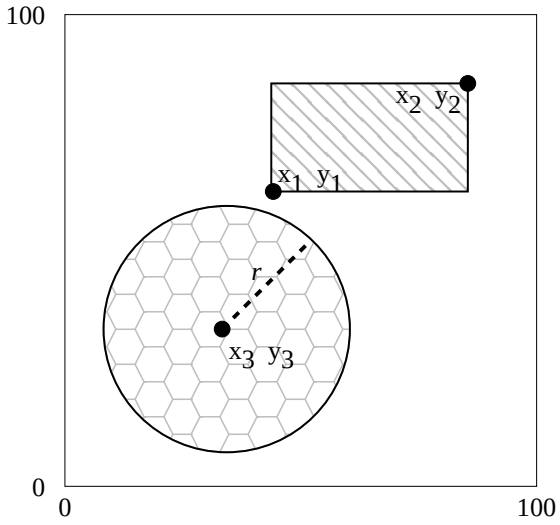
Two shapes are placed on a field. One is a circle, the other is a rectangle. You are interested in area of the field, extending from $(0, 0)$ to $(100, 100)$, that is covered by either shape. Any part of the shape extending beyond the field does not count, and any point is counted as covered if either or both shapes overlie that point.

You are required to calculate the total area rounded to the nearest integer value.

The first line of your input will tell you how many fields you are to calculate. Each subsequent line will consist of seven numbers, "x_1 y_1 x_2 y_2 x_3 y_3 r" each of which may have up to six decimal places of precision. These numbers describe the locations of the two shapes in a particular field. Each field is calculated independently.

These seven numbers have meanings as follows:

- x_1 is the x position of the bottom left corner of the rectangle
- y_1 is the y position of the bottom left corner of the rectangle
- x_2 is the x position of the top right corner of the rectangle
- y_2 is the y position of the top right corner of the rectangle
- x_3 is the x position of the centre of the circle
- y_3 is the y position of the centre of the circle
- r is the radius of the circle.



There should be an output line for each field calculated. As the field itself has $100 \times 100 = 10,000$ units of area, valid areas A are in the range $[0 \leq A \leq 10,000]$ remembering that $A \in \mathbb{I}$ (*i.e.*, the printed area is rounded to the nearest integer).

```
----- Sample Input ---
3
60 60 70 70 25 25 20
25 25 35 35 25 25 10
20 20 30 30 25 25 5
----- Sample Output ---
1357
336
100
-------end of input/output samples
```

PROBLEM E: Egg Experiments

You may have heard about experiments that we call "egg experiments" here. Given a number of eggs of the same characteristics and a building with a number of floors, the goal of an experiment is to find out the highest floor of the building from which we can drop an egg to the ground without breaking it. We call this floor number the "maximal safe floor". The experiment is conducted through a sequence of steps, where in each step we take one egg and drop it from a floor. We are limited by the number of the eggs we are given, since we need to find the maximal safe floor by the time all eggs are possibly broken. An egg which is dropped and not broken can be used again in the experiment, and we assume that its characteristics have not changed; *i.e.*, it has not become more "breakable" due to the drop. We call a "trial" the event of dropping an egg from a floor, and we want to perform the experiment with the minimal number of trials. Your programming task is to find the maximal number of trials that may be necessary with an optimal strategy for performing the experiment.

For example, if we have a building with 10 floors and just one egg, we obviously have to start dropping eggs from floor 1 and go up until it breaks, which can lead to at most 10 trials. Notice that the top floor can be the maximal safe floor. Of course, we could find this with only one drop from the 10^{th} floor, but if the egg breaks we would be in trouble. The lowest maximal safe floor may be 0, which means that the egg breaks even if dropped from the first floor. This means that even for a building with one floor, we still need to make one trial.

As another example, if we have two eggs and a building with 6 floors, we could drop the first egg from 3rd floor. If it breaks, we use two more trials with the remaining egg to test floors 1 and 2. If the first egg from floor 3 did not break, we could drop it from floor 5, and whether it breaks or not, we could use one more trial to test floors 4 or 6, respectively. This means that we can finish the experiment with a 6-floor building and 2 eggs with at most 3 trials.

These cases are recorded in the sample data.

Input

The input consists of a sequence of lines, where each line represents a test case and has two numbers separated by space: the number of floors of the building and the number of eggs. The number of floors is between 1 and 50000, and the number of eggs is between 1 and 20. The final line consists of two zeros, which denotes the end of the list of test cases.

Output

For each test case, your program should print one line with a number, which is the maximal number of trials that may be necessary with an optimal strategy to find the maximal safe floor.

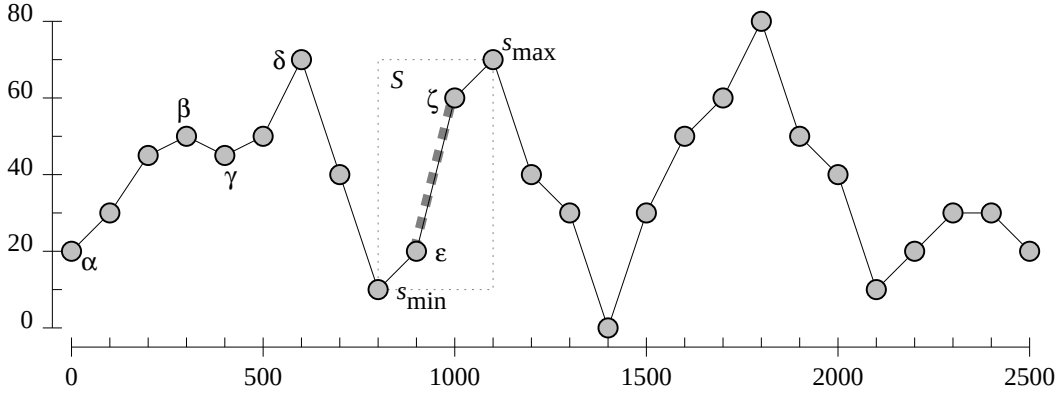
Sample input and output are provided on the next page.

Sample input and output for problem E

```
----- Sample Input 1 ---
1 1
10 1
6 2
0 0
----- Sample Output 1 ---
1
10
3
----- Sample Input 2 ---
1 1
0 0
----- Sample Output 2 ---
1
-------end of input/output samples
```

PROBLEM F: Alpine Hazard

You are presented with a list of altitudes along a path through the mountains. You are to calculate a "hazard number" for the path. The "hazard number" for the path is the height of the tallest segment of the path that contains the steepest slope.



A segment is defined as a monotonically increasing or decreasing sequence of altitudes (it can have a flat spot, but if it switches from going upwards to going downwards (or vice versa), that starts a new segment, even if the reversal is small). The first segment in the figure above therefore runs uphill from point α to point β and stops. It does not run all the way to point δ because between β and γ we went from going uphill to downhill.

If you consider the path shown above, you can see that steepest slope occurs between the points marked ϵ at $(900, 20)$ and ζ at $(1000, 60)$, so therefore the segment with the steepest slope is that within the box marked S, and the "hazard number" is the difference in height between the points marked s_{min} and s_{max}, or $70 - 10 = 60$.

If there are multiple segments with equally great maximum slope, the one with the greater difference in altitude is preferred.

Input

Your input will begin with an integer indicating the total number of paths. Each path should be solved separately. Each path begins with the word "`path`" on a line by itself, then has an integer indicating the number of points, and then a series of floating point numbers indicating the altitudes, one per line.

There will be 100 units of horizontal distance assumed to be between each altitude listed along the path, and you may assume that the full path has no more than 1000 altitudes recorded.

Output

For each path, you will output a single number rounded to two decimal places of precision indicating the "hazard number" for the path. Each hazard number will be placed on a separate line.

Sample input and output are provided on the next page.

Sample input and output for problem F

```
----- Sample Input ---
2
path
26
20.0
30.0
45.0
50.0
49.5
50.0
70.0
40.0
10.0
20.0
60.0
70.0
40.0
30.0
0.0
30.0
50.0
60.0
80.0
50.0
40.0
10.0
20.0
30.0
30.0
20.0
path
4
10.0
40.5
39.0
50.0
----- Sample Output ---
60.00
30.50
-------end of input/output samples
```

PROBLEM G: Secure Transmission

A government agent is assigned the mission to send certain documents, in groups, over a wireless channel without being completely intercepted by a hostile spying agent who is living within the range of his wireless channel.

Knowing that the spying agent is capable of listening to the wireless channel at any time and for a length of time T, once only without interruption per group of documents, the government agent decided to send their group of documents over N messages and hired you to write a program capable of sending all the messages in a certain order that minimizes the maximum number X of messages that can be entirely intercepted of a particular group. The documents content is considered disclosed if more than X out of the N messages of the group are intercepted entirely by the spying agent. Messages for a given group are sent sequentially in one continuous transmission. All input and output data are integer values.

The input file consists of:

1. The first line indicates the number of groups in the file.
2. Each group of documents is defined on four lines, consisting of:
   (a) An asterisk "$\star$" indicating the start of the group.
   (b) The next line contains N which represents the number of messages.
   (c) The third line contains N values representing the transmission durations of the messages.
   (d) The fourth line contains one value which represents T, the length of the listening period of the spy.

The output for each group of documents is simply the value of X, on a line by itself.
Each group is computed independently.

```
----- Sample Input ---
2
*
7
2 3 4 5 6 7 10
7
*
10
2 3 4 4 4 4 5 6 7 8
10
----- Sample Output ---
1
2
-------end of input/output samples
```

PROBLEM H: A Tack-y Situation

Ron will be sailing his ship to follow a series of buoys through some narrow channels. Your task is to compute the minimum number of times that Ron will need to turn his ship as he follows the buoys. Ron must touch each buoy. Do not count any turns at a buoy. The complication arises when Ron must travel upwind.

Ron's sailing ship cannot travel within 45 degrees of the oncoming wind. For example, if the wind is coming from the north then Ron cannot travel directly in a direction anywhere from north-west to north-east, including north. Instead, Ron must zigzag his ship in the given direction (called tacking) at angles nearest to his desired direction. Each corner of the zigzag requires a turn for Ron.

The first line of the file indicates the number of trips you will be computing (each one is done separately).

The first input line for a particular trip will specify the direction from which the wind is coming. It will be one of `n`, `ne`, `e`, `se`, `s`, `sw`, `w`, `nw` for the combinations of north (`n`), east (`e`), south (`s`) and west (`w`). So, "`ne`" is north-east – at 45 degrees between north and east. The next input line will specify the integer coordinates of Ron's starting point. The line after this will indicate the number of buoys, and each subsequent line for the trip lists the position of the next buoy (integer x, y coordinates) followed by the safe width of the channel that Ron can navigate (again as an integer) all separated by spaces.

Do not count turns at any buoy including the start and end buoys. Your output should be a single line in the form "`turns:` x" where x is the fewest number of turns possible to navigate the sequence of buoys.

You can assume that

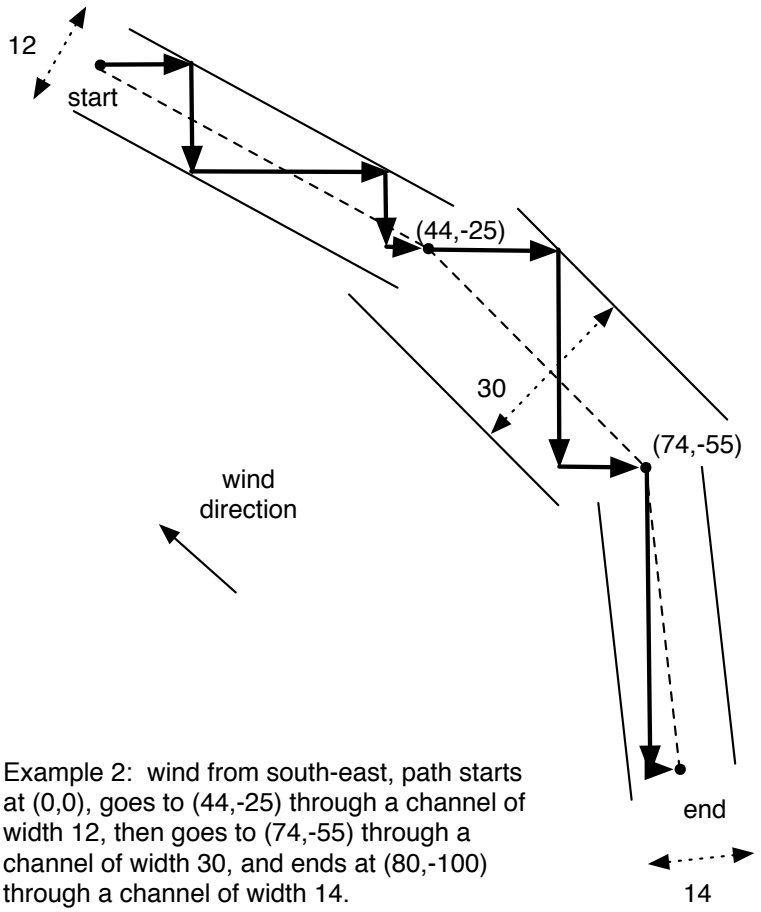
- no two consecutive legs of the buoy path will be collinear,
- the ship turns instantaneously on-the-spot, and
- all buoys including the start and end are in the centre of the channel.

To simplify round-off concerns in computation,

- no leg between buoys will follow exactly 45 degrees off the wind direction and
- each minimum-turn path between buoys will allow the ship to arrive at least one distance unit beyond the buoy.

The input below shows a data file with two trips. The first example trip is a single leg of travel from (0,0) to (0,35) against a north wind through a channel of width 20. The second example is formed of three legs of travel starting at (0,0) and ending at (80, -100) against a wind that comes from the south-east. There is a picture showing this second trip on the next page.

```
----- Sample Input ---
2
n
0 0
1
0 35 20
se
0 0
3
44 -25 12
74 -55 30
80 -100 14
----- Sample Output ---
turns: 2
turns: 7
-------end of input/output samples
```



Example 2: wind from south-east, path starts at (0,0), goes to (44,-25) through a channel of width 12, then goes to (74,-55) through a channel of width 30, and ends at (80,-100) through a channel of width 14.