Bandrosh's Contest

Francisco Leonardo Jales Martins

24/10/2007

Universidade Estadual do Ceará – UECE

Apoio: Joel Uchoa

Problem A: The Mysterious

(myst.{c,cc,java})

Description

One of the reasons for which École polytechnique (nicknamed “X" for reasons to be explained during the debriefing talk) is so deeply rooted in French society is its famous network of *camarades* – former students of the same school. When one camarade wants something (money, job, etc.), he can ask this network for help and support. In practice, this means that when he/she wants to reach some other camarade, not always of the same year, then surely he can find intermediate camarades to get to her/him. Note that the “camarade" relationship is symmetric. Due to the magic of the X network, there is always a means to reach anybody.

The program you have to write is supposed to help to minimize the number of these intermediate camarades.

Input / Output

The huge file of all living camarades is simplified so as to obey the following format. The first line in the file is the number of camarades, say N, an integer $1 \leq N \leq 10^5$. Camarades are labeled from 0 to N - 1. Follow N lines. Each line starts with the camarade label c, followed by the number of other camarades he/she knows, say nc, followed by the labels of those nc camarades. All these integers are separated by a single blank. It is assumed that nc is always less than 100. The last line in the file is the label of the camarade seeking help (say c1) followed by the label of the camarade he wants help from, say c2 (c2 != c1).

The input finish when *N* = -1.

Your program should output three integers separated by a blank: c1, c2 and the minimal number of intermediate camarades to reach c2. Have one blank line after of the each case test.

Sample Input :

```
4
0 3 1 2 3
1 1 0
2 2 0 3
3 2 0 2
1 2
-1
```

Sample Output:

```
1 2 1
```

Problem B: Divisibility

(divisibility.{c,cc,java})

On the planet Songa Monga, numbers are represented in base 62, using the digits
0, 1, . . . , 9, A, B, . . . , Z, a, b, . . . , z

where:

A (base 62) = 10 (base 10)
B (base 62) = 11 (base 10)
...
z (base 62) = 61 (base 10).

Given the digit representation of a number x in base 62, your goal is to determine if x is divisible by 61.

Input

The input test file will contain multiple cases. Each test case will be given by a single string containing only the digits '0' through '9', the uppercase letters 'A' through 'Z', and the lowercase letters 'a' through 'z'. All strings will have a length of between 1 and 10000 characters, inclusive. The end-of-input is denoted by a single line containing the word "end", which should not be processed. For example:

```
1v3
2P6
IsThisDivisible
end
```

Output

For each test case, print "yes" if the number is divisible by 61, and "no" otherwise. For example:

```
yes
no
no
```

Problem C: One by One

(one.{c,cc,java})

```
00000000000000000011
11111111000000000000
00000000000000001111
00000000000011000000
00000000000000111100
00000000000001110000
00111000000000000000
00000000000111000000
00000000111100000000
00000000000000000001
11000000000000000000
00001111111000000000
00000111111111111111
00000000011111100000
00000000001111111110
00000000000000011110
00000001111100000000
00000011111111110000
00011110000000000000
01111111111100000000
00000000000000000111
```

Operational Research is the law ,a time schedule is represented by a 0-1 matrix with n lines and m columns. Each line represents a person and each column an event. All the persons participating to an event have a one in the corresponding entry of their line. Persons not attending the event have a zero entry in that column. Events occur consecutively.

Problem

Write a program that finds a smart permutation of the events where each person attends all its events in a row. In other words, permute the columns of the matrix so that all ones are consecutive in each line.

input

The first line of the input consists in the number $n \le 400$ of lines. The second line contains $m \le 400$, the number of columns. Then comes the n lines of the matrix. Each line consists in m characters 0 or 1. The input matrix is chosen so that there exists only one smart permutation which preserves column 0 in position 0. To make things easier, any two columns share few common one entries. The input has finish when n = -1 .

Output

The output consists of m numbers indicating the smart permutation of the columns. The first number must be 0 as column 0 does not move. The second number indicate the index (in the input matrix) of the second column, and so on. Have one blank line after of the each case test.

Sample input

```
3
4
0110
0001
1101
6
5
01010
01000
10101
10100
00011
00101
21
20
00101000000000000000
10010010010110010100
00101101000000000000
01000000000000001000
00000101100000100000
01000000100000100000
00000010000110000000
01000000000001001000
00000000001001000011
00001000000000000000
10000000000000000100
00010010011000010011
01111101111001111011
01000000000001101011
01100101100001101001
00100101100000000000
00010000001001000011
01010000101001111011
00000010010010010000
00010010011111010111
00101001000000000000
-1
```

Sample output

```
0
3
1
2

0
2
4
3
1

0
17
11
12
6
9
15
3
10
18
19
13
16
1
14
8
5
7
2
4
```

Problem D: Clothes Store

(store.{c,cc,java})

In the laundry next to my at, clothes are stored on coat hangers that are put on hooks fixed on a circular rail moved electrically by a computer. Hooks are numbered so that finding a cloth is easy. The rail moves in front of a mark.

Details

We model the rail as an array of dimension N, referenced in a circular way, that is indices are to be considered modulo N. When a batch of n clothes must be stored, the launderer types the number n on a keyboard. The computer then looks for the first location of n+2 free hooks, from the current position of the rail to the right, yielding zone k::k +n+1 (all indices considered modulo N). Once this is done, the rail moves so that hook numbered k+n+1 arrives on the mark, and the launderer puts the n clothes on the hooks k+1::k+n. Hooks k and k+n+1 are not used to store clothes, but are used as \separators" between batches of clothes. The launderer then gives the ticket number k to the customer. Hooks used to hang the clothes of some customer are assigned to the customer, even during the actual cleaning of the batch of clothes.

When the customer comes back with the ticket number k, the launderer types k on the keyboard and the computer makes the rail moves so that the separating hook k of the corresponding batch is in front of the mark. The launderer takes the batch back (during this operation, the rail does not move) and gives it back to the customer. When a cloth is handed back to a customer, the corresponding hook is then free. Note that, when both its left and right neighbors are empty, a separating hook can be used for any purpose (either to hang a cloth or to become a separating hook again)

The aim of the program is to model deposits and withdrawals of batches of clothes.

An input file has the following format. The first line contains the number N of hooks, (1 ≤ N ≤ 300). We then have the number l of lines in the file after the current one. Follow l lines with two different possible formats. **The input finish when N = -1**. The first one is:

D n
to deposit n clothes. The second one is:
W k
to indicate that clothes corresponding to ticket k must be withdrawn (0 ≤ k < N) .

When the customer makes a deposit of clothes, the program looks for an empty place for the *whole batch*. If this cannot be found, the program's output is

No space left, please come back later.

If ticket k can be issued, the program's output is

The launderer gives ticket k.

When ticket k is given back, the program's output is

The launderer gives back batch k.

and all hooks used to hang the clothes of the corresponding customer are made free. More over, a separating hook of a batch that has been removed is also made free if both its right and left neighbors are free.

Whenever hooks h, . . . ,h+q become free, the program should output

i is freed.

for all i between h and h+q.
We assume that at the beginning of the reading, the rail is empty and that hook 0 is in front of the mark. Only clothes that have been deposited can be withdrawn. Have one blank line after of the each case test.

Sample Input :

```
22
5
D 1
D 3
W 0
D 3
D 11
7
7
D 2
D 1
W 0
D 4
D 1
W 5
D 1
-1
```

Sample Output :

```
The launderer gives ticket 0.
The launderer gives ticket 2.
The launderer gives back batch 0.
0 is freed.
1 is freed.
The launderer gives ticket 6.
The launderer gives ticket 10.

The launderer gives ticket 0.
The launderer gives ticket 3.
The launderer gives back batch 0.
0 is freed.
1 is freed.
2 is freed.
The launderer gives ticket 5.
No space left, please come back later.
The launderer gives back batch 5.
6 is freed.
0 is freed.
1 is freed.
2 is freed.
The launderer gives ticket 5.
```

Problem E: Dungeons & Dragons

(d&d.{c,cc,java})

Description

In the Role Play Game of Dungeons & Dragons, players often roll multi-sided dice to generate random numbers which determine the outcome of actions in the game. These dice come in various flavors and shapes, ranging from a 4-sided tetrahedron to a 20-sided isocahedron. The faces of an n-sided die, called dn for short, are numbered from 1 to n. Ideally, it is made in such a way that the probabilities that any of its n faces shows up are equal. The dice used in the game are d4, d6, d8, d10, d12, and d20.

When generating random numbers to fit certain ranges, it is sometimes necessary or desirable to roll several dice in conjunction and sum the values on their faces. However, we may notice that although different combinations of dice yield numbers in the same range, the probabilities of rolling each of the numbers within the range differ entirely. For example, a d6 and a d10 afford a range of 2 to 16 inclusive, as does two d8s, but the probability of rolling a 9 differs in each circumstance.

Your task in this problem is to determine the probability of rolling a certain number, given the set of dice used.

Input

The input test file will contain multiple cases, with each case on a single line of input. The line begins with an integer d (where $1 \leq d \leq 13$), the number of dice rolled. Following are d descriptors of dice, which can be any of “d4”, “d6”, “d8”, “d10”, “d12”, or “d20”. The last number on each line is an integer x (where $0 \leq x \leq 1000$), the number for which you are to determine the probability of rolling with the given set of dice. End-of-input is marked by a single line containing 0; do not process this line. For example:

```
1 d10 5
2 d6 d6 1
2 d10 d6 9
2 d8 d8 9
0
```

Output

For each test case, output on a single line the probability of rolling x with the dice, accurate to five decimal places. Note that even if there trailing zeros, you must show them (see Test problem for an example of decimal formatting). For example:

```
0.10000
0.00000
0.10000
0.12500
```

Problem F: Box of Glass

(box.{c,cc,java})

Description

You are given a three-dimensional box of integral dimensions *(ℓx X ℓy X ℓz).* The edges of the box are axis-aligned, and one corner of the box is located at position *(0, 0, 0).* Given the coordinates *(x, y, z)* of some arbitrary position on the surface of the box, your goal is to return the square of the length of the shortest path along the box's surface from *(0, 0, 0)* to *(x, y, z).*

Examples

If $\ell_x = 1$, $\ell_y = 2$, and $\ell_z = 1$, then the shortest path from (0, 0, 0) to (1, 2, 1) is found by walking from (0, 0, 0) to (1, 1, 0), followed by walking from (1, 1, 0) to (1, 2, 1). The total path length is $\sqrt{8}$.

Input

The input test file will contain multiple test cases, each of which consists of six integers $\ell_x, \ell_y, \ell_z, x, y, z$, where $1 \leq \ell_x, \ell_y, \ell_z \leq 1000$. Note that the box may have zero volume, but the point (x, y, z) is always guaranteed to be on one of the six sides of the box. The end-of-file is marked by a test case with $\ell_x = \ell_y = \ell_z = x = y = z = 0$ and should not be processed. For example:

```
1 1 2 1 1 2
1 1 1 1 1 1
0 0 0 0 0 0
```

Output

For each test case, write a single line with a positive integer indicating the square of the shortest path length. (Note: The square of the path length is always an integer; thus, your answer is exact, not approximate.) For example:

```
8
5
```

Problem G: Planets

(planet.{c,cc,java})

Description

In a fictitious solar system Véia , consisting of star S, a planet P, and its moon M, planet P orbits in a perfect circle around star S with a revolution period of exactly T Earth days, and moon M orbits in a perfect circle around planet P with an unknown revolution period. Given the position of moon M relative to star S at three different time points, your goal is to compute the distance of planet P from star S.

To do this, consider a two-dimensional Cartesian coordinate system with its origin centered at star S. You may assume that P's counterclockwise orbit around S and M's counterclockwise orbit around P both lie completely within the x–y coordinate plane. Let (x_1, y_1) denote the position of the moon M on the first observation, let (x_2, y_2) denote its position k_1 Earth days later, and let (x_3, y_3) denote its position k_2 Earth days after the second observation.

Input

The input test file will contain multiple test cases. Each test case consists of two lines. The first line contains the integers T , k_1, and k_2, where $1 \le T, k_1, k_2 \le 1000$. The second line contains six floating-point values x_1, y_1, x_2, y_2, x_3, and y_3. Input points have been selected so as to guarantee a unique solution; the final distance from planet P to star S will always be within 0.1 of the nearest integer. The end-of-file is denoted with a single line containing "0 0 0".

```
360 90 90
5.0 1.0 0.0 6.0 -5.0 1.0
0 0 0
```

Output

For each input case, the program should print the distance from planet P to star S, rounded to the nearest integer.

```
5
```

Problem H: Roller Coaster

(roller.{c,cc,java})

Description

The super cows are building a roller coaster! They want your help to design as fun a roller coaster as possible, while keeping to the budget.

The roller coaster will be built on a long linear stretch of land of length L (1 ≤ L ≤ 1,000). The roller coaster comprises a collection of some of the N (1 ≤ N ≤ 10,000) different interchangeable components. Each component i has a fixed length Wi (1 ≤ Wi ≤ L).Due to varying terrain, each component i can be only built starting at location Xi (0 ≤ Xi ≤ L-Wi). The cows want to string together various roller coaster components starting at 0 and ending at L so that the end of each component (except the last) is the start of the next component. Each component i has a "fun rating" Fi (1 ≤ Fi ≤ 1,000,000) and a cost Ci (1 ≤ Ci ≤ 1000). The total fun of the roller coster is the sum of the fun from each component used; the total cost is likewise the sum of the costs of each component used. The cows' total budget is B (1 ≤ B ≤ 1000). Help the cows determine the most fun roller coaster that they can build with their budget.

Input Format

* Line 1: Three space-separated integers: L, N and B.
* Lines 2..N+1: Line i+1 contains four space-separated integers,
respectively: Xi, Wi, Fi, and Ci.

Sample Input:

```
5 6 10
0 2 20 6
2 3 5 6
0 1 2 1
1 1 1 3
1 2 5 4
3 2 10 2
```

Output Format

* Line 1: A single integer that is the maximum fun value that a
roller-coaster can have while staying within the budget and
meeting all the other constraints. If it is not possible to
build a roller-coaster within budget, output -1.

Sample output:

```
17
```

Problem I: Big Brother

(bb.{c,cc,java})

A spy agency wants to monitor all communications in a computer network. They have a budget for at most 10 installations of spying software on 10 of the host computers on the network. For the software to work properly each communication line *A*–*B* must have at least one host *A* or *B* being monitored.

Input will consist of a number of network scenarios. Each scenario will contain:

• An integer *n* ($10 \leq n \leq 2500$) on a line on its own, giving the number of hosts in the network.

• A line of data for each host (thus *n* lines in total) giving the list of other hosts to which the host has a direct communication line. The hosts are named 0, 1, . . . , $n - 1$; the first line of data gives the neighbours of host 0, the second those of host 1, and so on up to host $n - 1$.

Each of these lines consists of an integer d ($1 \leq d < n$) which is the number of neighbours host *k* has, followed by *d* integers which are the neighbouring hosts' names, separated by spaces. The neighbours will be given in numerical order, and will each be valid host names in 0, 1, . . . , $n - 1$ other than *k*. (Note that each of these input lines may thus be up to 2500×5 characters in length.)

The last line of input will be a '0' on a line by itself. This line should not be processed.

Output will consist of one line for each input network, indicating whether the network can be successfully spied upon by infecting 10 nodes. Each line of the output will consist of '**Network** *n*: ', where n is the scenario number, starting at 1, followed by '*yes*' or '*no*'.

Sample Input:	Sample Output:

```
11
5 1 3 5 8 10
5 0 2 4 6 9
5 1 3 5 6 10
4 0 2 4 6
4 1 3 5 7
5 0 2 4 6 8
6 1 2 3 5 7 9
4 4 6 8 10
4 0 5 7 9
4 1 6 8 10
4 0 2 7 9
11
10 1 2 3 4 5 6 7 8 9 10
10 0 2 3 4 5 6 7 8 9 10
10 0 1 3 4 5 6 7 8 9 10
10 0 1 2 4 5 6 7 8 9 10
10 0 1 2 3 5 6 7 8 9 10
10 0 1 2 3 4 6 7 8 9 10
10 0 1 2 3 4 5 7 8 9 10
10 0 1 2 3 4 5 6 8 9 10
10 0 1 2 3 4 5 6 7 9 10
10 0 1 2 3 4 5 6 7 8 10
10 0 1 2 3 4 5 6 7 8 9
12
11 1 2 3 4 5 6 7 8 9 10 11
11 0 2 3 4 5 6 7 8 9 10 11
11 0 1 3 4 5 6 7 8 9 10 11
11 0 1 2 4 5 6 7 8 9 10 11
11 0 1 2 3 5 6 7 8 9 10 11
11 0 1 2 3 4 6 7 8 9 10 11
11 0 1 2 3 4 5 7 8 9 10 11
11 0 1 2 3 4 5 6 8 9 10 11
11 0 1 2 3 4 5 6 7 9 10 11
11 0 1 2 3 4 5 6 7 8 10 11
11 0 1 2 3 4 5 6 7 8 9 11
11 0 1 2 3 4 5 6 7 8 9 10
0
```
	```
	Network 1: yes
	Network 2: yes
	Network 3: no
	```