IDI Open Programming Contest April 14th, 2007

The Problemset

Tips

- Tear the problem set apart and share the problems among you.
- Problems are not ordered by difficulty.
- Try solving the easy problems first. Two problems in this set are tagged with “easy” to help you getting started.
- If you get “incorrect answer” on a problem, you can print your program and debug it on paper while you let someone else work on a different problem on the computer.
- Contact Truls or Nils if you need help.

Rules

- Each team consists of one to three contestants.
- One computer is used per team.
- You may not cooperate with persons not on your team.
- You may print your programs on paper to debug them.
- What you may bring to the contest:
  - Any written material (Books, manuals, handwritten notes, printed notes, etc).
  - Pens, pencils, blank paper, stapler and other useful non-electronic office equipment.
  - NO material in electronic form (CDs, USB pen and so on).
  - NO electronic devices (PDAs and so on).
- The only electronic content you may consult during the content is that specified by the organiser (see the web-page). You may not copy source code from web pages, etc.
- Your programs should read from standard in and write to standard out. Writing to standard error will result in a failed submission. C programs should return 0 from `main()`.
- Your program may use at most 100MB of memory.
- Your programs may not:
  - access the network,
  - read or write files on the system,
  - talk to other processes,
  - fork,
  - or similar stuff.
  - If you try, your program will hang or crash. If it hangs, it will take a couple of minutes before others will be able to run their programs. And please do not crack somebody who uses their spare time trying to give you something valuable.
- Show common sense and good sportsmanship.

Problem A

Help Chelsea! (easy)

When football clubs achieve poor results, there is only one thing to do: Buy new players! This is the most popular strategy among the major football clubs in Europe today, and Rosenborg is an example of a Norwegian club which has used this strategy with success. They have many talent scouts travelling around the earth to find promising young footballers.

Chelsea struggles in Premier League at the moment, and they have decided to buy another player. But they are sick and tired of waiting around for a talent scout to find a descent player, and employ a way more efficient strategy. They actually put a famous saying from Bærum into practice: "If something is on sale, you can be sure there is a reason why."

In a football setting, this means that the most expensive player is probably the best one. Hence, looking for a new player only involves calling all football clubs and asking for their most expensive player. Your task is to help Chelsea find the most expensive player from a list.

Input specifications

The input has $n \leq 100$ cases, where n is given by the first line of input. The first line of each test case is a single positive integer, $p \leq 100$, giving the number of players to consider. Then follow p lines, where each line represents a player. The line starts with a positive integer $c_i < 2 \cdot 10^9$, the price of player i. Then follows a single space before the name of the player. All player prices are unique. Player names are never more than 20 characters long, and contain no spaces.

Output specifications

For each test case your program should output a single line giving the name of the most expensive player.

Sample input

```
2
3
10 Iversen
1000000 Nannskog
2000000 Ronaldinho
2
1000000 Maradona
999999 Batistuta
```

Output for sample input

```
Ronaldinho
Maradona
```

Problem B

Virus

Some mad chemistry dude has made a new atomic virus bomb that will kill everybody. It works like this:

The virus has N different forms. Every second, a virus in one form will transform into one or more viruses of the same or different forms. Furthermore, each form will produce a certain number of tritium atoms during the second it takes to transform. When the total number of such atoms reaches the critical limit L, they explode as a hydrogen bomb.

We want to know how long time it takes from a poor guy gets infected until he blows up.

A simple example would be a virus with only one form, which will transform into two viruses of the same form every second, and will produce one tritium atom in one second. Let's say that $L = 15$. After one second there are two viruses and one atom. After two seconds there are four viruses and three atoms. After three seconds there are eight viruses and seven atoms. After four seconds the bomb goes off.

Let's look at a little more complex example with a virus with two forms A and B, and $L = 500$. Every second a virus of form A will transform into three viruses of form A and one virus of form B, and produce one tritium atom. A virus of form B will transform into two viruses of form B, and produce one hundred tritium atoms. If we start with one virus of form A, then after one second we have 3 A, 1 B, and 1 atom. After two seconds we have 9 A, 5 B, and 104 atoms. After three seconds we have 27 A, 19 B, and 613 atoms. So the answer is three seconds.

Input specifications

The first line of the input gives the number of test cases $T \leq 100$. The first line of each test case contains $1 \leq N \leq 20$ and $1 \leq L \leq 1000000000$. Then follow N lines, one for each form. The description of a form contains $N + 1$ non-negative integers less than 1000. The first N describe how many viruses of the different forms it transforms into. The last describes how many tritium atoms it produces in the process.

Output specifications

Output one line for each test case. We assume that the patient is infected with one virus of the first form. If the virus bomb will never go off, print "`lucky`". Otherwise, print the number of seconds it takes before the patient blows up.

Sample input

```
3
1 15
2 1
2 500
3 1 1
0 2 100
1 15
2 0
```

Output for sample input

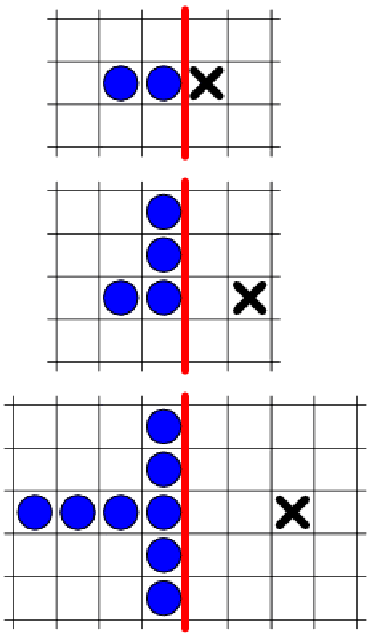
```
4
3
lucky
```

Problem C

Frogger

A group of frogs is sitting on an infinite xy grid. The frogs all have non-positive x coordinates. On the grid location $(2,0)$ there's a fly. The frogs are hungry and would like to get there to eat the fly. There's one catch; a frog can only move by jumping over another frog sitting next to it, and then the frog being jumped over explodes. So the frog John sitting at $(-1,0)$ can jump to $(1,0)$ if Peter sits at $(0,0)$, but then Peter explodes and is gone. Similarly, if Alfred sits at $(-1,2)$ and Barney sits at $(-1,1)$, then Alfred can jump to $(-1,0)$ and Barney explodes.

So getting to the fly is going to have a high cost in frog lives, but the frogs all believe in the common good and are willing to sacrifice everything for the sake. The question however is whether they can make it at all. Given the location of the fly on the non-negative x axis, how many frogs do you need to start with? The frogs can initially be placed anywhere on or to the left of the y-axis, but must have distinct locations.



Input specifications

The first line of the input gives the number of test cases $T \le 100$. For each test case, there's one line with a single integer $0 \le X \le 31$, meaning that the fly is located at $(X,0)$.

Output specifications

For each test case, output one line with the number of frogs you need to start with to catch the fly, or "`frogger`" if it's not possible to do it.

Sample input

```
4
0
1
2
3
```

Output for sample input

```
1
2
4
8
```

Problem D

Conquistador

Rune Johan is a nice young boy and a clever student, but he has had little luck engaging persons of the female persuasion in fruitful conversation. He was very grateful when the computer science department organised a ball, and invited the first grade nurse students. There he saw *her*, the woman of his dreams. But as most of us do in such situations, he froze up, and did not dare to approach her.

After a few days of regretting his lack of action, Rune Johan decides to make up for it. He finds her name, Celina Middleware, but does not want to contact her before he has made a decent plan. He starts by locating one of her friends, Bjørgfrid, and invites her to a cup of coffee. After a few minutes of conversation, Bjørgfrid tells that she has some trouble with her computer. Rune Johan makes a deal with her. He will fix her computer if she tells him everything he needs to know about Celina.

It turns out that Celina likes many different kinds of boys, but she is still rather picky. He could be intelligent, cultivated and well dressed, or own a motorcycle and be slightly rude. Or he could simply be rich. Rune Johan writes down all the information, and goes home to finish his plan. He makes an estimate of how many weeks it would take to fulfil each of the criteria, and tries to decide which combination takes less time. He thinks that he can work on all of them in parallel.

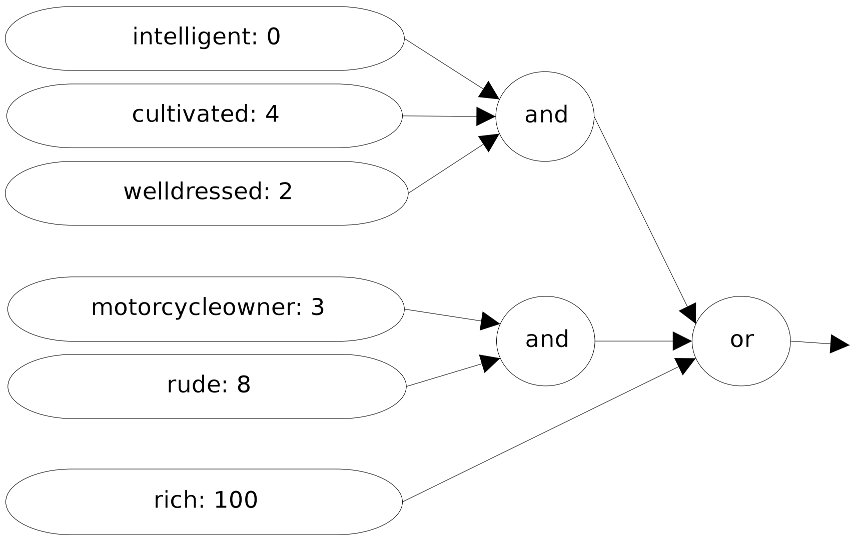


Figure 1: What Rune Johan must become to conquer Celina Middleware.

Input specifications

The first line of input gives the number of test cases, which is at most 100. Each test case is given as two strings on separate lines. The first string gives the time costs of fulfilling the criteria, separated by commas. Each cost is given by a string giving the name of the criterion, a colon, and then the integer time cost of the criterion in weeks. The names contain symbols 'a' to 'z', and their length is from 1 to 20. The costs are between 0 and 1000 inclusive. There will be no more than 20 different criteria.

The second string gives the combinations of criteria which will satisfy miss Middleware. Each combination is separated by the symbol '&', and the criteria are separated by the symbol '|'. There will be between 1 and 10 combinations in each scenario. Each combination will contain at least one criterion, and name no criterion more than once.

Output specifications

For each test scenario, output a line with the minimum time cost in weeks for satisfying Celina's desires.

Sample input

```
3
intelligent:0,cultivated:4,welldressed:2,motorcycleowner:3,rude:8,rich:100
intelligent&cultivated&welldressed|motorcycleowner&rude|rich
ab:13,b:17,cab:21
ab&b|b&cab
a:14,b:13,c:14,d:11
a&b&c|d&a&c|a|b&d
```

Output for sample input

```
4
17
13
```

Problem E

Party

Kjell Bratbergsengen, head of the computer science department, is concerned about the small number of girls studying computer science at NTNU. Representatives from Norwegian software companies complain constantly, but Kjell believes that the most dangerous possible effect of this problem is low overall recruitment in the future. It is well known that geeks usually do not have parents who studied at Dragvoll, and Kjell wants to do all he can to make sure his current students gets the opportunity to reproduce with decent partners.

While considering how to cope with this problem, Kjell remembers how things were when he studied at Gløshaugen. He recalls a lot of cute nurse students. It was not always easy to connect with them, as they were usually more interested in medical doctors than geeks, but charming young Kjell had his share of success nonetheless. Kjell believes that todays students also have a chance, and decides to arrange a party. Some male students already have girlfriends, but Kjell wants to make sure that all the single students get a date for the party as well. He tries to recruit nurse students, but realizes that it is impossible to get the number he needs, and many of them are picky about who they want date. How can he get a date for all the single geeks?

After some thinking, he comes up with the solution: He can just arrange several parties with the same girls! It is expensive to arrange parties, so there should be as few as possible. Even though Kjell is a talented programmer, he is now very busy with administrative chores, and needs your help writing a program which decides the minimum number of parties he will have to host.

Input specifications

The input has $n \leq 200$ cases, and the first line consists of a positive integer giving n. The first line for each test case consists of two positive integers separated by a single space, $m \leq 100$ and $f \leq 50$, where m denotes the number of male students who need a date, and f the number of nurse students available.

Then follow f lines. Line number i represents nurse number i. The line starts with a positive integer giving the number of male students she is willing to date. Then follows a list of space separated unique integers naming these geeks. The male students are numbered from 0 to $m-1$.

Output specifications

Output one line for each test case. If it is impossible to make sure all male students get a date, no matter how many parties Kjell arranges, output a line with the text "`impossible`". Otherwise output a line with a single integer giving the minimum number of parties needed to make sure all male students can attend a party with a date.

Sample input

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

Output for sample input

```
2
3
impossible
```

Problem F

Save the computer!

Life as a computer science student is hard. Apart from the curriculum related challenges, one also has economical ones. Some of the funding needs to be spent on food, rent, and so on, but we all agree that it is way more important to have enough money to make sure your precious computer is always up and running.

Even though the funding from Lånekassen is more or less evenly distributed over the semester, you have realized that except for the first month you need all the money you get for food and rent. You thus have to get your computer budget from the first month, and have a certain sum available. If you do not use this sum the first month, you will surely waste it on gadgets, so you better spend it wisely on computer equipment as soon as you get the money.

As we all know, a computer consists of several components. If any one of these fails, the computer fails. If so happens, you of course have the discomfort of having to find something else to do, and you also have to deal with constant teasing from your friends while your PC is down. It is obvious that you need to plan ahead to avoid this embarrassing situation.

You realize that you should use your computer budget on spare components. In that way, if a component fails, you can replace it immediately, and your computer will work again. Components fail at different rates, and they also have different prices. Your task now is to maximise the probability that your computer will work the whole semester, by deciding how many spares of each component you should buy within your restricted budget.

In order to solve this task, you have to model the rate of failure of the different components in your computer. This is usually done with a Poisson distribution:

$$P_i(k,t) = \frac{e^{-\lambda_i t}(\lambda_i t)^k}{k!}$$

$P_i(k,t)$ is the probability that component i will fail exactly k times during t time units. We will only look at this problem for one semester at a time. The variable t can thus be set to 1 permanently, and the equation reduces to:

$$P_i(k) = \frac{e^{-\lambda_i}\lambda_i^k}{k!}$$

λ_i is the expected number of times component i will fail in one semester. You may assume that the probability that one component fails is independent of the failure of other components.

Input specifications

The input has $n \leq 50$ cases, and the first line consists of one positive integer giving n.

The input for one test case consists of 3 lines. The first line contains two positive integers separated by a single space, $1 \leq c \leq 500$ and $0 \leq b \leq 500$. c is the number of components in your computer that may fail, and b is the size of your computer budget.

Next follows one line with c floating point numbers in double precision. The ith number denote the expected number of times component i will fail in one semester, $0.0 \leq \lambda_i \leq 5.0$. The last line of each test case consists of c positive integers. The ith number denotes the price of component i, $1 \leq r_i \leq 100$.

Output specifications

For each test case, you should output a single line with the maximum probability of survival you can achieve, with 5 digits precision.

Sample input

```
2
2 3
0.5 0.3
3 1
3 10
0.8 0.0 0.2
5 3 2
```

Output for sample input

```
0.67399
0.80786
```

Problem G

Fridge of Your Dreams (easy)

Eirik drinks a lot of Bingo Cola to help him program faster, and over the years he has burned many unnecessary calories walking all the way to the kitchen to get some. To avoid this he has just bought a small fridge, which is beautifully placed next to his computer. To make it match his fancy big-tower with all its blinking LEDs, it is necessary to style it a bit.

He has bought a weight sensor with a display and a small general purpose programmable chip, to put underneath the fridge. The idea is to make the display show how many litres of Bingo Cola there is in the fridge. To do this he must read a binary register in the sensor, and convert it to a decimal number to be displayed.

Input specifications

The first line of input gives $n \leq 1000$, the number of test cases. Then follow n lines with positive numbers represented as 24-bit binary strings (`0`s and `1`s).

Output specifications

For each number, output its decimal representation, without any leading zeros.

Sample input

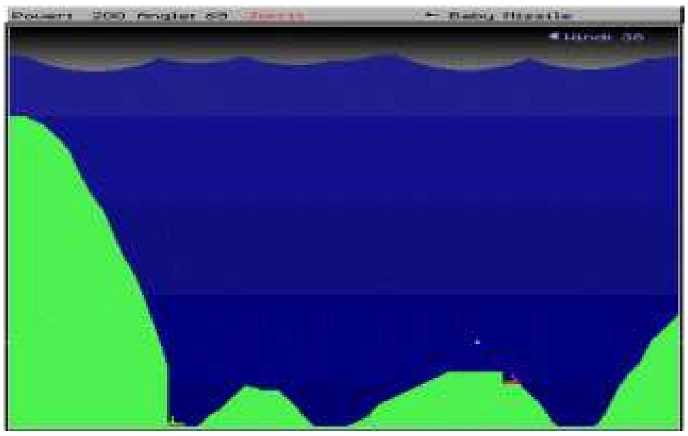
```
5
000000000000000000000001
000000000001010101010101
000000000000000000001010
101011001010101100101101
111111111111111111111111
```

Output for sample input

```
1
5461
10
11316013
16777215
```

Problem H

Scorched Earth



The problem has been slightly simplified after the contest to fix some bugs and difficulties. The original problem had $0 \leq d \leq 180$ *and no limitation* $x_u < x_o$.

General Arne Heisveis is a victim of the constant cuts in funding for the Norwegian defence. There is basically no money available, and Arne is forced to spend his workdays in a dull office doing nothing.

To avoid an unworthy death from boredom, one of Arne's colleagues found an old computer game called Scorched Earth, which the generals now play all day. The problem is that Arne is not very talented in this game, but still very competitive. He therefore wants you to write a program to help him cheat.

A screen-shot from Scorched Earth is shown above. The generals' contest will be a series of battles between two players, and the only allowed weapons are small missiles. Each player controls a tank, and the objective is to destroy the opponents tank by shooting it. The players take turns in shooting, and control the angle and initial velocity of their missiles. The initial velocity can never exceed 300.0 m/s, and can of course never be negative.

In order for a projectile to hit, it must avoid all the mountains in the field, and have the correct velocity and angle to hit the opponent. The gravity is always 9.8 m/s^2, and there may also be wind. To keep things simple, we assume that the wind gives the projectile a constant acceleration.

Arne is quite confident in finding an angle that will avoid all the mountains in the battlefield, but needs your help adjusting the velocity of the shot.

Input specifications

The input has $n \leq 1000$ cases, where n is given by the first line of input. Each test case is described by a line with 6 floating point numbers x_u, y_u, x_o, y_o, w, d. Your tank is positioned at (x_u, y_u) in meters, and your opponents at (x_o, y_o), where $0.0 \leq x_u < x_o \leq 1000.0$ and $0.0 \leq y_u, y_o \leq 800.0$. The number $-2.0 \leq w \leq 2.0$ gives the acceleration in m/s^2 of the projectile along the x-axis caused by the wind. The angle chosen by Arne is given by $0 < d < 78$ in degrees. An angle $d = 0$ implies a shot along the increasing x-axis, and $d = 90$ would have implied a shot along the increasing y-axis.

Output specifications

Output for each test case a line with an initial velocity within the bounds which will ensure a hit, with 5 digits precision. If this is not possible, output a line with the text "`impossible`".

Sample input

```
2
0.0 0.0 500.0 0.0 0.0 45.0
100.0 0.0 500.0 0.0 0.0 135.0
```

Output for sample input

```

70.00000
impossible
```

Problem I

Free Willy

Willy is sitting behind bars in Alcatraz. Jan Erik Vold is guarding him, and gives him a challenge:

”I managed to transform KULTURUKE into UKTURKULE by applying these permutations in succession:

bcdefaghi cabfdeghi bcadefghi adcefgbhi cgabdefhi cdaefhgbi

That’s what gave me the start of my great poem: KULTURUKE ULTURKUKE TULKURUKE ULTKURUKE UKTURULKE TLUKURUKE UKTURKULE

Now, I want you to do the transformation using the same set of available permutations. I permuted 6 times, but if you can manage to do it by permuting fewer times than I did, then I’ll unlock the cage!”

”That’s easy, I only need 4 permutations!” says Willy, ”You first apply bcadefghi to get ULKTURUKE. Then cdaefhgbi to get KTUURKULE. Then bcadefghi again to get TUKURKULE. And finally bcadefghi a third time to get UKTURKULE.”

”Oh, you’re not a big, dumb fish after all” says Jan Erik and brings out the keys. Willy jumps into the ocean and lives happily ever after!

Input specifications

The first line of the input gives the number of test cases $T \leq 30$. The first line of each test case contains $1 \leq N \leq 26$, $1 \leq P \leq 10$, and $1 \leq L \leq 10$. The second line contains two words with N characters each. Then follow P lines, each with an allowed permutation of the first N letters of the alphabet (in lowercase).

Output specifications

For each test case, output one line with the minimum number of times you need to apply one of the allowed permutations to the letters of the first word in order to arrive at the second word, or “`whalemeat`” if it’s not possible to do it in at most L steps.

Sample input

```
3
9 6 5
KULTURUKE UKTURKULE
bcdefaghi
cabfdeghi
bcadefghi
adcefgbhi
cgabdefhi
cdaefhgbi
9 5 4
kulturuke tlukuruke
bcdefaghi
cabfdeghi
bcadefghi
adcefgbhi
cgabdefhi
9 3 4
WILLFREEY FREEWILLY
bacdefghi
abghefdic
fecdbaigh
```

Output for sample input

```
4
whalemeat
4
```