IDI Open
Programming Contest
April 21st, 2012

Problem Set

A Boss Rush
B Decode the Message (Easy)
C Troublesome Tools
D Civilization (Easy)
E Inverse Divisor Sums
F Longest Common Path
G Birthday Party
H Holey Road
I Candy Store
J Rotating Penguin Maze

Jury and Problem Writers

Ruben Spaans, NTNU
Jon Marius Venstad, NTNU
Geir-Arne Fuglstad, NTNU
Magnus Lie Hetland, NTNU

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 point you in the right direction.
- If your solution fails 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.
- If you need help, contact the judges.

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

Boss Rush

You are playing the fantastic new game Mega Man XIII-2. Unfortunately, the boss fights in the game are extremely hard. You have access to several weapons, like the PowerLaser, the EvilRocket and the PsychTerror. Each weapon belongs to one of three categories: laser, rocket and psiotic. In order to defeat a boss you must use three weapons, one from each category. A given boss is only vulnerable against a subset of the weapons, but you have studied the bosses in advance and know which weapons you can use against each of them.

You start the game with **two** of every available weapon. As the game progresses, you encounter every boss in a predefined order. For each boss fight, you choose one weapon from each category, and after the fight those three weapons are worn out and cannot be used again.

In order to progress as far as possible into the game, you need to select your weapons carefully. You don't want to lose against the last boss because you already spent the needed weapons in the beginning of the game! However, you are overwhelmed by the number of ways you can pick weapons against the bosses, so you have decided to write a computer program that determines the maximum number of bosses you can defeat if you pick the weapons for each boss fight optimally.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case begins with a line containing a single integer N, the number of bosses in the play session. Then N groups of 3 lines follow, a total of $3N$ lines. The i-th group of 3 lines describes which weapons can be used to defeat boss i from the categories laser, rocket and psiotic, respectively (one line for each category). These lines start with an integer W_{ij}, the number of weapons from the j-th category that can be used to defeat boss i. The remainder of a line contains W_{ij} space-separated strings containing the names of the weapons.

Output specifications

For each test case, output on a single line M, the maximal number of bosses you can beat.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 100$
- $0 < W_{ij} \leq 10$
- All string lengths are between 1 and 32, inclusive.
- A string only consists of upper- and lowercase letters from `a` to `z`.

Sample input

```
1
4
3 UltraLaser TurboLaser PowerLaser
2 EvilRocket ShatterRocket
3 PsychTerror Wisp Shutdown
2 UltraLaser TurboLaser
2 ShatterRocket PropelledCannon
1 Wisp
2 UltraLaser TurboLaser
2 PropelledCannon EvilRocket
1 Wisp
3 UltraLaser TurboLaser PowerLaser
2 EvilRocket ShatterRocket
1 Wisp
```

Output for sample input

```
3
```

Problem B

Decode the Message (Easy)

Two of your friends Alex and Adam have developed a secret system for sending messages to each other. The only problem is that the system has turned out to be too hard to do by hand. Therefore, Alex has asked for your help to write a program to decode the secret messages. After making you swear never to reveal the system, he explains that the coded message consists of one or more words, only consisting of lower case characters `a`–`z`, seperated by spaces. Each word corresponds to one character, `a`–`z` or space, and is found by adding the value of each character in the word and taking the remainder when dividing by 27. `a` is assigned the value 0, `b` is assigned the value 1 and so on up to `z` which is assigned the value 25. Based on the calculated remainder the values 0–25 corresponds to `a`–`z` as before, and the value 26 corresponds to a space.

Help your friend write a program which takes a coded message and outputs the decoded message.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case consists of a single line containing a secret message. The secret message contains only lower case characters and spaces, and there are only spaces between words and never two or more consecutive spaces.

Output specifications

For each test case, output the corresponding decoded message.

Notes and Constraints

- $0 < T \leq 100$
- Each secret message contains no more than 1000 characters

Sample input

```
2
a b c d e f
it is late o green wrong whole
```

Output for sample input

```
abcdef
a horse
```

Problem C

Troublesome Tools

The gentleman Inco Gnito is attempting to infiltrate the Entership Starprise, and has been tasked with assisting the chief engineer Forgie with repairs. Forgie has brought with him some tools, and will repeatedly ask Inco for a subset of these. Only when Inco has placed exactly the correct tools in front of Forgie will he take them, utilise them and return them to Inco. This process will repeat itself until the repairs are complete. This sounds simple enough, but the problem is that Inco has no idea what any of the tools are called! Luckily Inco is a quick study, and he is thus able to use all the information of his previous errors and successes when trying to find a new subset of tools. On the other hand, if he is very unlucky, he could be stuck with Forgie for a very very long time, and possibly be revealed as a spy.

Calculate how many different subsets of tools Inco will have to offer Forgie in total, assuming he has the least possible amount of luck during this assignment.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case begins with a line containing two integers N, the number of tools, and K, the number of different subsets Forgie will ask for. The next line contains N space-separated tool names consisting of up to 25 lowercase letters 'a'-'z' and '-'. Then follow K lines containing an integer M and then the names of M different tools as above. *Tool names are unique within a test case, and the tools for different test cases are not related.*

Output specifications

For each test case, output the maximal number of subsets Inco might have to offer Forgie, assuming he learns all he can during the assignment. *The answer can be a very large number, so output the result modulo* $2^{31} - 1$.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 1000$
- $0 < K \leq 100$
- $0 < M \leq N$

Sample input

```
2
3 3
tricorder quantum-flux-regulator hyperspanner
1 tricorder
1 quantum-flux-regulator
1 hyperspanner
5 4
pulse-drill phase-coil-resonator duct-tape crowbar optronic-coupler
2 phase-coil-resonator optronic-coupler
2 pulse-drill optronic-coupler
1 crowbar
1 duct-tape
```

Output for sample input

```
6
19
```

Problem D

Civilization (Easy)

Emperor Montezuma of the ancient civilization of the Aztecs has a hard time planning the productivity of his capital city. The city is divided into square-shaped regions of the same size. Each region has three important properties: Workforce (number of people able to work), income (the amount of money Montezuma can earn from this region, given in Quetzal, the Aztec currency) and the number of farms. A region needs an administrator in order to contribute to the city. Otherwise, the region is self-sufficient and not officially a part of the city (and hence, Montezuma doesn't earn taxes, cannot utilize the work force and cannot access food produced from the region's farms).

Montezuma wants to make sure that his capital city is sufficiently prosperous, and he wants to achieve this prosperity by using as few administrators as possible. The capital city is prosperous if the sums of the workforces, income and farms for all administered regions exceed or are equal to the given values in each category.

Alas, Montezuma was born rather early and didn't have access to computers. However, you have read about Montezuma's exploits in the Civilopedia of the latest installment of the computer game Civilization. You found this scenario interesting enough that you have decided to write a computer program that calculates the fewest number of regions Montezuma would need to administer in order to have a city which is prosperous with regard to workforce, taxes and farms.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case begins with a line containing a single integer N, the number of regions near the capital city. The next line contains three integers W, C, F, representing the number of workers, tax income and number of farms which is needed for the city to be considered prosperous. The next N lines contains three integers w_i, c_i, f_i, the number of workers, the tax income and the number of farms that region i offers, respectively.

Output specifications

For each test case, output the minimal number of regions that need to be administered. If it is impossible for the city to be prosperous, output "`game over`" instead (without quotes).

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 18$
- $0 < W, C, F \leq 1000$
- $0 < w_i, c_i, f_i \leq 1000$

Sample input

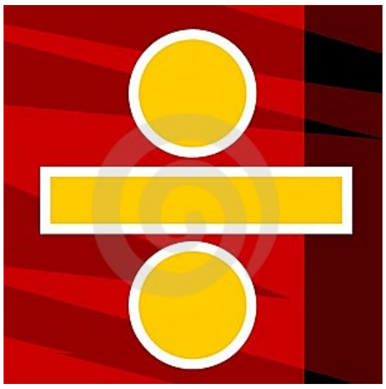
```
2
3
50 50 50
10 20 30
40 30 22
10 10 33
2
10 20 30
5 5 5
200 10 20
```

Output for sample input

```
2
game over
```

Problem E

Inverse Divisor Sums



Your friend Odd Even is obsessed with number theory. He likes to learn new operations to perform on numbers, and to spend endless hours applying his newly acquired knowledge to numbers. For instance, last year he learned about Euler's totient function $\phi(n)$, which counts the number of positive integers less than or equal to n that are relatively prime to n. Shortly after that, he went on a spree and calculated $\phi(n)$ for all integers n from 1 up to one million by hand.

Recently, he learned that the sum of all divisors of a number N could be calculated by the following formula:

$$\text{sum of divisors}(N) = \prod_{i=1}^{r} \frac{p_i^{(a_i+1)} - 1}{p_i - 1}.$$

Here, $p_1^{a_i} p_2^{a_2} \dots p_r^{a_r}$ is the factorization of N into prime factors where each p_i is different and a_i is the maximum power of p_i such that $p_i^{a_i}$ that divides N.

Odd Even wants to calculate the function in reverse; given a positive integer N he wants to find all positive integers M having N as its sum of divisors, and he wants them written out nicely in increasing order. You think this will take too long, so you have decided to intervene and offer computer assistance.

Write a computer program that does the following: given a positive integer N, output a list of all the integers M having N as its sum of divisors, in increasing order, or inform Odd Even that no such numbers exists.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. The T following lines each contain a single integer N.

Output specifications

For each test case, output all such numbers on one line, in increasing order, with a single space between each number. If no numbers exist, output "`none!`" (without quotes).

Notes and Constraints

- $0 < N \leq 10^9$
- The output can be large. For Java, it is advisable to buffer the output using StringBuilder.

Sample input

```
4
7
2
126
1524
```

Output for sample input

```

4
none!
68 82
704 1083 1523
```

Problem F

Longest Common Path

Per and Pål are friends, and like to hang out together. They are also very impatient, so they always take the shortest possible way when they need to go somewhere.

It's the end of the school day, and they need to go home. Naturally, they want to arrive at home as early as possible. They also want to maximize the time they walk together while walking towards their homes. You have been asked to write a computer program that calculates the maximal time they can walk together. Their neighbourhood is modelled as a graph where intersections are nodes and roads are edges. The roads are bidirectional, and it takes the same time to traverse a road in both directions. All important destinations happen to be located at an intersection. Also, the school and the two homes are never located at the same intersection. There exists a path between every pair of intersections.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. The next line contains two integers N and M, the number of intersections and the number of bidirectional roads in the city, respectively. The following line contains three integers S, P and Q, indicating the position of the school, Per's home and Pål's home, respectively. The next M lines contain three integers a_i, b_i and t_i, indicating that there is a bidirectional road between intersections a_i and b_i which takes c_i minutes to traverse. All intersections are numbered from 0 to $N-1$, inclusive.

Output specifications

For each test case, output the length of the longest distance Per and Pål can walk together while still arriving at their own homes as fast as possible.

Notes and Constraints

- $0 < T \leq 100$
- $3 \leq N \leq 2000$
- $N - 1 \leq M \leq \min\left(\frac{N(N-1)}{2}, 10000\right)$
- $0 \leq a_i < b_i < N$
- $0 < c_i \leq 1000$
- $0 \leq S, P, Q, a_i, b_i < N$

Sample input

```
2
4 5
0 2 3
0 1 100
1 2 50
1 3 40
0 2 500
0 3 500
4 5
0 2 3
0 1 100
1 2 50
1 3 40
0 2 10
0 3 10
```

Output for sample input

```
100
0
```

Problem G

Birthday Party

N persons have been invited to a somewhat special birthday party. Each person brings one present, but the recipent of each present is determined randomly. A person never receives his own present, but all other persons are equally likely recipients. What is the probability that one can find k persons at the party such that person 1 gives his present to person 2, person 2 gives his present to person 3 and so on to person k which gives his present to person 1?

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case consists of two integers N and k.

Output specifications

For each test case, output the probability with an accuracy of at least 10^{-6}.

Notes and Constraints

- $0 < T \leq 30$
- $2 \leq N \leq 10000000$
- $2 \leq k \leq N$

Sample input

```
4
2 2
3 2
3 3
10 3
```

Output for sample input

```
1.000000000
0.750000000
0.250000000
0.313469843
```

Problem H

Holey Road

You are manouvering a tiny remote-controlled car along a road in very bad condition; the road is full of holes. The car is unable to drive over the holes, since it would plunge into the hole and become damaged beyond repair. In addition, driving off the road will cause the car to be forever lost in the tall grass surrounding it.

The car is so small that it can be considered as a point with no spatial extension. The road is W meters wide and L meters long, and it runs parallel to the y-axis. Your car starts at $\left(\frac{W}{2}, 0\right)$ and your destination is $\left(\frac{W}{2}, L\right)$. All of the holes happen to be perfectly circle-shaped, and none of them intersect or touch other holes or the edges of the road.

You want to take the shortest possible path to the destination. Write a program that calculates the length of such a path.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case begins with a line containing three integers N, W, L, the number of holes, and the width and the length of the road, respectively. Then follow N lines each containing three integers x_i, y_i, r_i representing a hole with center x_i, y_i and radius r_i.

Output specifications

For each test case, output the length of the shortest possible path from the starting position to the final position that avoids holes. An error of up to 10^{-6} will be accepted in the output.

Notes and Constraints

- $0 < T \leq 100$
- $0 \leq N \leq 100$
- $0 < L \leq 1000$
- $0 < W \leq 100$
- $r_i < x_i < W - r_i$
- $r_i < y_i < L - r_i$
- $0 < r_i < \min\left(\lfloor \frac{W}{2} \rfloor, \lfloor \frac{L}{2} \rfloor\right)$

Sample input

```
3
1 50 1000
20 500 5
1 50 1000
25 500 5
3 100 1000
50 50 25
25 150 24
75 150 24
```

Output for sample input

```
1000.00000000000
1000.05000041668
1009.34797846036
```

Problem I

Candy Store

Little Anne loves candy and she wants to buy candy at the local candy store. Fortunately, she also happens to be extremely rich, which enables her to buy all the candy she could ever want. She wants to buy candy for at least C kroner, and she also wants to buy at most one of each type.

There are many ways to buy candy for at least C kroner, so she wants to consider the possibilities before buying. Since her parents think she is too young to use a computer, she has asked you to write a program that calculates the number of possibilities.

Anne doesn't like large numbers like 1,000,000,007, so you instead calculate the number of possibilities modulo 65537, a less intimidating number.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case begins with a line containing two integers N and C, the number of available candy types and the least amount of money Anne wants to spend. The next line contains N space-separated integers a_i, the cost of candy type i in kroner.

Output specifications

For each test case, output on its own line the number of ways Anne can buy candy for at least C kroner, modulo 65537.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 200$
- $0 < C \leq 10000$
- $0 < a_i \leq 200$

Sample input

```
2
5 10
5 6 7 8 9
10 100
10 10 10 10 10 10 10 10 11 9
```

Output for sample input

```
26
1
```

Problem J

Rotating Penguin Maze

Sometimes I just need to manouver a baby penguin as quickly as possible through a maze, so it can find the fish someone has hidden inside. I have taught the penguin to be proficient with a compass, and can thus give it instructions 'E', 'N', 'W' and 'S'. The penguin's maze also has an extra feature: at some of the tiles are pressure plates that cause a 90 degree counter-clockwise rotation of the whole iceberg on which the maze is located, when the penguin steps on one of them. The penguin is oblivious to this, as it rotates along with the iceberg. The compass needle, however, doesn't rotate along with the iceberg, and thus I have to modify my instructions whevenever such a pressure plate is activated. In the long run, I think this will give me a headache, so please, can you write a program to help me out?

Fortunately the maze always has exactly one path from the starting tile to the goal tile. Between each pair of tiles there exists exactly one path, and the starting tile will never contain a pressure plate.

Input specifications

The first line of the input consists of a single integer T, the number of test cases. Each test case begins with a line containing two integers X, Y - the size of the maze in the X and Y directions, and another line containing 4 integers px, py, gx, gy - the penguin's starting coordinates and the goal tile coordinates. Then follow Y lines with X characters each, where each character represents a single tile of the labyrinth as a base-32 number $(0, 1, \ldots, 9, \mathrm{A}, \mathrm{B}, \ldots, \mathrm{V})$. This number is the sum of the features the tile contains: If the tile has an east-going passage, 1 is added to the sum. If the tile has a north-going passage, 2 is added. For west-going passages 4 is added, and for south-going ones 8. In addition, if the tile contains a pressure plate, 16 (or G) is added to the sum. Thus e.g. a tile value of H (or 17) means the tile has an east-going passage and a pressure plate.

Output specifications

For each test case, output a single line with the compass instructions you will have to give your penguin to guide it along the shortest path to the tile where the fish can be found, assuming each instruction makes it move exactly one tile in the given direction.

Notes and Constraints

- $0 < T \leq 100$
- $0 < X \leq 500$
- $0 < Y \leq 500$
- $0 \leq px, gx < X$
- $0 \leq py, gy < Y$

Sample input

```
1
2 3
0 0 1 2
9C
AQ
22
```

Output for sample input

```
ESE
```