IDI Open
Programming Contest
April 2nd, 2011

Problem Set

Jury and Problem Writers

Eirik Reksten, Steria

Ruben Spaans, NTNU

Erik Axel Nielsen, McKinsey & Company

Tor Gunnar Høst Houeland, IDI/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

Soundex

Soundex is a phonetic algorithm for transforming a string into a code, which is always a letter followed by three digits. The purpose is that strings with a different spelling but similar pronounciation will be transformed into the same code. The transformation rules are as follows:

- The first letter of the string is the first letter of the code.
- Subsequent consonants are replaced by digits:
  - `b`, `f`, `p`, `v` with 1
  - `c`, `g`, `j`, `k`, `q`, `s`, `x`, `z` with 2
  - `d`, `t` with 3
  - `l` with 4
  - `m`, `n` with 5
  - `r` with 6

  `h` and `w` are ignored. The vowels, `a`, `e`, `i`, `o`, `u` and `y`, are not encoded.
- Two or more adjacent letters with the same digit are replaced with a single digit. Two or more letters with the same digit separated with `h` or `w` are also replaced by a single digit. Two letters with the same digit separated by a vowel will appear as the digit twice.
- Repeat the previous step until it isn't possible to replace repeating digits with one digit.
- If the resulting code has less than 3 digits, pad the end with zeroes until the code has 3 digits. If there are more than 3 digits, drop the rightmost digits.

Some example transformations are:

- Both `robert` and `rupert` are transformed to R163.
- `baawwwww` is transformed to B000.
- `hopp` is transformed to H100. The first letter of the string always becomes the first letter in the code, even if it is a vowel or `h` or `w`.
- `ratatata` is transformed to R333, because the vowel between each pair of `t` (3) forces the digit to be repeated.
- `yhhhwthwhtwhthwhwth` is transformed to X300. All occurrences of `h` and `w` are ignored, leaving only one `3` in the code.
- `bbpb` is transformed to B100. The first `b` is kept separate from the last three `b`'s. The remaining `b`'s are consecutive and are replaced with one digit.

Your evil friend Halvor has noticed that a lot of different words will give the same soundex code. For example, the strings `rhhhbm`, `rubeno`, `rpowam`, `robnew` and 73908 other strings with 6 or less letters will be converted to the code R150. This makes him very curious, and he wants to know the number of strings of the a given length or shorter that will be converted to the same soundex code. Naturally, he wants you to write a computer program to accomplish this task. Your friend does not care about upper or lower case, so `AA`, `Aa`, `aa` etc are considered to be equal strings and should only be counted once.

Input specifications

The first line of input contains a single integer T, the number of test cases to follow. Each test case begins with a line containing a string S and an integer number L. S represents a soundex code, and consists of one uppercase letter followed by three digits. L represents the maximum length of the original string which is converted into the given soundex code.

Output specifications

For each test case output one line containing a single number, the number of strings of length L or less having the soundex code S. This number can be large, so output it modulo 1,000,000,007.

Notes and Constraints

- $0 < T \leq 100$
- $0 < L \leq 1000$
- All soundex codes will start with an uppercase letter.
- Two strings S and T are equal if they have the same length and the letters s_i, t_i at each position are equal, regardless of case.
- Only letters between `a` and `z` are considered. No `æ`, `ø`, `å` or other non-English characters are to be used.
- The soundex code is always legal. That is, a non-zero digit will never follow a zero digit, and the digits range from `0` to `6`.

Sample input

```
3
A300 2
R150 6
X123 1
```

Output for sample input

```
2
73912
0
```

Problem B

Sheep Frenzy

Since IDI Open '09, I've had lots of sheep counting programs supposed to help me fall asleep. Sadly, none of these were any good, so I've spent close to every night the last years cursing those evil animals. Not sleeping gives you quite a bit of spare time, and I've spent mine creating a new game called Sheep Frenzy.

The player controls Ulgr the Unpleasantsmelling, whose objective is to eat all sheep on each level as fast as possible. Strangely enough, though I've gotten quite good at it, this still doesn't give me the satisfaction of revenge on the evildoers. This is why I need your help. You need to write a program that calculates the best way of moving Ulgr around the board in order to eat all sheep as fast as possible.

The board is organized in a $H \times W$ grid, where each cell is one of the following:

- 'U' - The starting point of Ulgr the Unpleasantsmelling. He likes eating sheep, and you're gonna help him do so.
- '#' - Sheep. They are obviously blind, deaf and has no sense of smell, as they won't move even when you come to eat them.
- '.' - Grass. Don't argue, it is grass.
- 'X' - Mountains. You (Ulgr) can't walk on these tiles. Well, to be honest you can, but please don't. The game locks up if you do.

Ulgr makes one move each second. One move consists in eating a sheep or moving one cell to the left, right, up or down. He can only eat a sheep if he stands on the same cell as it.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case begins with a line containing two numbers, H and W, the height and width of the grid for that level. Then follow H lines, each containing W characters, describing that part of the grid.

Output specifications

For each test case, output a line containing a single number, the minimum time in seconds Ulgr needs to eat all the sheep on that level. If it is not possible to eat all the sheep, output `impossible` instead.

Notes and Constraints

- $0 < T \leq 100$
- $0 < H, W \leq 50$
- There will always be at least one and never be more than 16 sheep in one test case.
- Each test case will contain exactly one 'U'.
- All mountains on the board are marked with 'X'. That is, neither Ulgr or any sheep are on a mountain.

Sample input

```
2
2 2
U.
.#
3 5
#..X#
..XXX
.U...
```

Output for sample input

```
3
impossible
```

Problem C

LOL (Easy)

Your friend Andreas has a very good sense of humour. In fact, it is so good that he even enjoys to change words so that they will contain the substring `lol`. For example, during the 2010 FIFA World Cup in soccer he had much fun changing the word `fotball` to `fotbalol`. In order to be more efficient in his word plays, he has asked you to make a computer program that finds the minimum number of changes needed in order to transform a string into a new string containing `lol` as a substring. There are three legal ways to change a string: delete a character, insert a new character and replace an existing character with a new one.

Input specifications

The first line of the input consists of a single number T, the number of test cases. Then follows T lines, containing a string S consisting of lower case letters between `a` and `z` only.

Output specifications

For each string, output on its own line the minimum number of changes needed in order to obtain a string containing `lol` as a substring.

Notes and Constraints

- $0 < T \leq 100$
- $0 < |S| \leq 50$ (That is, the maximal string length is 50.)

Sample input

```
4
fotball
sopp
ingenting
spillolje
```

Output for sample input

```

1
2
3
0
```

Problem D

Treasure Hunt

Your boss is probably the richest man on earth. He loves games and fun, and has just invested in a new island for the sole purpose of enjoyment for himself and his wealthy friends. His current plan is to arrange a huge treasure hunt, and the task of spreading the treasure around the island fell on you.

After months of walking around on the island and digging down treasure, your boss has more news for you. Apparently the island is an old mine field. As he (obviously) cannot risk the lives of himself and his friends, he tells you to start disarming the mines. "You're very lucky", he tells you, "we know the location of these mines".

Despite the happy news, you quickly realize that this is something you probably won't succeed at. After all, when the second mine is disarmed, chances are you don't have any limbs left. Disarming the rest with no limbs is not easy. Therefore, you sit down and think hard about a solution that might solve your problem. The next day, you go to your boss with the following proposal.

Instead of disarming all the mines on the island, you suggest making a fence that separated the mines from the rest of the island. Then the treasure hunt could be held on the safe side of the fence.

Your boss is unsure. First of all, he would require the fence to be entirely straight, due to esthetic reasons. He's also not sure whether there will be enough treasure within such a setup to justify his treasure hunt. So now you need to figure out how much treasure you can isolate with such a straight fence. Being a skilled programmer (a hobby of yours) you sit down and write a program that calculates how much treasure you can possibly separate from the mines with such a straight fence.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case begins with a line containing two numbers, N and M, the number of treasures and the number of mines, respectively. Then follow two lines containing N integers, describing the locations of the N treasures. Integer i on the first line describes the x-coordinate of the i-th treasure, while integer i on the second line describes its y-coordinate. Then follow two more lines, each containing M integers. They describe the x- and y-coordinate of the mines, respectively, in the same manner as the preceding lines did for the treasures.

Output specifications

For each test case, output a line containing a single number, the maximum number of treasures you can isolate using the strategy above.

Notes and Constraints

- $0 < T \leq 100$
- $1 < N \leq 4000$
- $1 < M \leq 4000$
- $0 \leq x_i, y_i \leq 1000000$
- The fence can be modeled as an infinitely thin, infinite straight line (ie. assume that the island is convex).
- Mines or treasures cannot lie exactly on the fence/line.
- There will be no treasures and/or mines at the exact same point.

Sample input

```
2
2 3
1 3
1 1
1 2 3
2 1 2
3 3
1 3 5
1 1 1
1 2 3
2 1 2
```

Output for sample input

```
1
2
```

Problem E

Cross Country Race

The organizers of the world championship in interval start cross country have made a fatal blunder. It is impossible to pass the person in front of you anywhere along the race tracks. If one person catches up with the one that started before him/her, they will have to stay together for the rest of the race.

This is obviously no just way of declaring a world champion, so the organizers decide to only record the times of the people who reach goal without someone right in front of them. The rest will run another race (when they have rested), to decide their race time.

The danger of this setup is that you will have to arrange too many races. Given the length of the race, and the speeds of the participants, how many races will they need to arrange?

In the race, there is one minute between each starter, in the order given in the input. In subsequent races, the relative order between the participants is the same, but their starting times will be adjusted so that it is still exactly one minute between each start.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case begins with a line containing two numbers, N, the number of participants in the championship, and S, the length of the race, in meters. Then follow a single line containing N numbers $v_1, v_2, ..., v_N$, the speeds of the participants in the order they will start the race, given in meters/minute.

Output specifications

For each test case, output a line containing a single number, the amount of races that will have to be arranged.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 1000$
- $0 < S \leq 50000$
- $0 < v_i < 1000$
- For the purposes of this problem, each participant is considered a single point.
- A participant can reach the exact point of the previous starter, but not pass it.
- For the purposes of this problem, a participant has someone right in front of them if and only if they are at the exact same point, and he did not start first of them.
- The relationship between the distance s, the velocity v and the time t can be expressed as $s = v * t$.

Sample input

```
2
5 100
20 20 20 20 20
4 2000
100 200 300 350
```

Output for sample input

```
1
3
```

Problem F

Beads

A game consists of putting beads in boxes. The rules of the game are too complex to describe here, but all you need to know is that keeping track of the number of beans in adjacent boxes are very important to the outcome of the game.

You are asked by a friend to write a program to help him win the game every time. At the start of a game, all boxes are empty.

Input specifications

The first line of the input consists of a single number T, the number of games played. Each game start with a line describing B, P and Q, the number of boxes, put requests and query requests, respectively.

Then follows $P + Q$ lines with either `P` i a, saying a beads are put in box number i, or `Q` i j, a query request for the number of beads in boxes i through j, inclusive.

Output specifications

For each query request, output the number of beads in boxes a through b, inclusive, that are in the boxes at this point of the game.

Notes and Constraints

- $0 < T \leq 100$
- $0 < B \leq 100000$
- $0 < P \leq 30000$
- $0 < Q \leq 30000$
- $0 \leq a \leq 100$
- $0 < i \leq j \leq B$
- Note that boxes are 1-indexed.
- This is an I/O-heavy problem. For Java programmers, this means that you should use `BufferedReader` for input reading (not `Scanner`). It is also beneficial to build all output in a `StringBuilder` before printing in a single print statement.

Sample input

```
1
7 5 3
P 2 1
P 3 3
P 4 7
Q 1 4
P 7 6
Q 4 4
P 6 4
Q 1 7
```

Output for sample input

```
11
7
21
```

Problem G

Sleeping at Work

You are at work. Unfortunately, there is a programming competition immediately after your working hours. In order to perform well, you need some sleep at work to regain as much energy as possible. Your workday is N minutes long, and each minute has an energy value, $e_0, e_1, \ldots, e_{N-1}$. Your sleep requirement is exactly M minutes, but you can only sleep for a maximum of R minutes in a row before your boss notices. There is a bonus if you sleep for several minutes in a row; the i-th minute in a row will have its energy value multiplied by i. For instance, if you sleep for three minutes having energy values of 10, 10 and 9, you will gain $10 + 2 \cdot 10 + 3 \cdot 9 = 57$ energy. After you have slept for M minutes, you are fully rested and cannot sleep any more that day. You have decided to write a computer program which calculates the maximum amount of energy you can gain during a given workday.

Input specifications

The first line of input contains a single integer T, the number of test cases to follow. Each test case begins with a line containing three integer numbers, N, the number of minutes in your workday, M, the sleep requirement in number of minutes, and R, the maximum number of minutes in a row you can sleep before your boss notices. Then follows a line containing N numbers, $e_0, e_1, \ldots, e_{N-1}$, each minute's energy value.

Output specifications

For each test case output one line containing a single number, the highest amount of energy that can be gained by sleeping a total of M minutes, or output `impossible` if it is not possible to get the required amount of sleep.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 500$
- $0 < M \leq 50$
- $0 < R \leq 50$
- $0 \leq e_i \leq 100$
- You can only start and stop sleeping exactly when the minute indicator on the clock changes.

Sample input

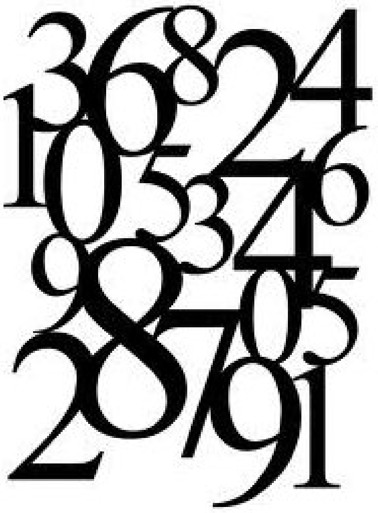
```
2
10 3 3
10 10 9 6 5 4 2 1 4 4
10 6 1
1 2 3 4 5 6 7 8 9 10
```

Output for sample input

```
57
impossible
```

Problem H

Is it a Number? (Easy)



This is it! You've finally graduated and started working. Looking forward to some really cool tasks now. While you're skipping around in the eagerness of getting started, you're told what your first task is - Input Validation! You should check whether the typed input is an integer number.

Time to get going! Given a sequence of characters, check whether they describe an integer number. Whitespace is allowed both before and after the number, but the rest of the input must consist of a single, non-negative integer number. Only digits will be accepted as the relevant part of the input (`+` is not allowed, for instance).

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Then follow a single line for each test case; the input to be validated.

Output specifications

For each test case, output a line containing the value of the number if the input is a valid integer number, or `invalid input` (all lowercase) if the input is not.

Notes and Constraints

- $0 < T \leq 500$
- Each test case will consist of at least 1 and at most 50 characters (excluding the line break).
- A test case can contain any character with an ASCII value between 32 and 126 (inclusive).
- There should be no leading zeros in the output.

Sample input

```
4
 23 456
-36
  0045
 44.3
```

Output for sample input

```
invalid input
invalid input
45
invalid input
```

Problem I

Proud Penguin

Proud Penguin (PP) is one of the hottest attractions in your city. Specializing in the arctic area, they let you see fish, seals, whales and, of course, penguins. The penguins being such a huge success, PP has decided to expand with a whole new area for the penguins to play around on. This new area is shaped as a long, narrow track, consisting of climbs and slides (with the height being highest at the ends, so that a travel always starts with a slide). The penguins will be allowed to enter on one side, and then travel to the other side by waddling, swimming and sliding. Connecting the two current areas, this will make the life of the penguins a lot more varied. Of course, the glass on the one side of the track will provide visitors with a lot more penguin action as well.

PP is now faced with one last problem in the planning stage of the expansion. How should they distribute water along the track? Due to the lazy nature of the penguins, they have decided to go for the solution where the highest climb will be as low as possible. At the same time, the board has decided to cut maintenance costs, and have set a maximum limit on the amount of water to be used.

You are appointed with the task of finding an optimal water distribution along the track. Given the height of the track at evenly spaced intervals and the maximum amount of water you can use, what is the lowest possible maximum climb you will have to leave the penguins with?

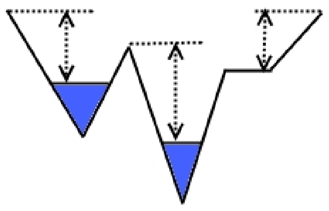


Figure 1: Measuring the height of climbs

Input specifications

The first line of input contains a single integer T, the number of test cases to follow. Each test case begins with a line containing two integer numbers, N, the length of the track for that test case, and W the amount of water available. Then follow a line containing $N+1$ integers a_i, where a_0 describes the height at the left side, a_1 the height one unit from the left side, and so on until number a_N, which describes the height at the right side of the track.

Output specifications

For each test case output one line containing a single number, the height of the lowest possible maximum climb for the track in that test case.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 10000$
- $0 \leq W \leq 1000000$
- $0 \leq a_i \leq 100$
- $a_0 = a_N = 100$
- In your calculations, assume that the width of the track is always one unit, and that the track between two points is a straight line.
- A climb starts at water level or at any land point, and continues upwards until no adjacent point is higher than the current one (ie. a strictly increasing path).
- The height of a climb is defined as the height difference between its lowest and highest point.
- Keep in mind that the penguins will want to travel in both directions.
- Assume that water always runs to a lower adjacent point.
- An error of up to 10^{-6} will be accepted in the output.

Sample input

```
2
2 0
100 34 100
5 25
100 70 90 60 75 100
```

Output for sample input

```
66
19.573186408981247
```

Problem J

Travelling Tom

Tom is a used popsicle salesman. His recent sales at his stand on Gløshaugen haven't been very good, so he's decided that he wants to travel the world selling his merchandise. He has already compiled a list of cities to visit, and have also gathered the costs of the plane trips between them. Now he just have to figure out if he has enough money for the trip. He has of course heard that this travelling salesman problem is really really hard, so the task is down to the smartest people he knows. That's you (if you don't feel very smart at the moment, keep in mind that Tom doesn't know that many people). You need to write a program that computes the lowest cost of Tom visiting all these cities in the given order.

Input specifications

The first line of input contains a single number T, the number of test cases to follow. Each test case begins with a line containing a single number, N, the number of cities Tom will visit. The second line of each test case contains N numbers a_i, the order in which Tom wants to visit the N cities. He starts his trip in the first city described, and will travel back there once he has visited the last one. Then follow N lines, each containing N integers, describing the cost c_{ij} of travelling from city i to each of the N cities.

Output specifications

For each test case, output a line containing a single number, the lowest possible cost of visiting the cities. If it is not possible to visit all the cities, output `impossible` instead.

Notes and Constraints

- $0 < T \leq 100$
- $0 < N \leq 200$
- $0 \leq a_i < N$
- $-1 \leq c_{ij} \leq 10000$
- The list of cities visited will always be a permutation of the numbers from 0 to $N - 1$, inclusive.
- The cost of travelling from a city to itself is always 0.
- If and only if there is no plane going from city i to city j, then the j-th number of the i-th line of costs will be -1.
- Note that Tom is returning to the starting city at the end of the tour.
- Cities can be visited several times, but will only count when (also) in the correct relative order. For the order `1 0 2`, for instance, `1 2 0 2 1` is a valid tour, while `1 2 0 1` is not.

Sample input

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

Output for sample input

```
5
impossible
```