ANZAC Round 1 – 2018

September 1, 2018

Contest Problems

This contest contains eleven problems over 24 pages. Good luck.

For problems that state "*Your answer should have an absolute or relative error of less than* 10^{-6}", your answer, x, will be compared to the correct answer, y. If $|x - y| < 10^{-6}$ or $\frac{|x-y|}{|y|} < 10^{-6}$, then your answer will be considered correct.

Definition 1

For problems that ask for a result modulo m:
If the correct answer to the problem is the integer b, then you should display the unique value a such that:

- $0 \leq a < m$
  and
- $(a - b)$ is a multiple of m.

Definition 2

A string $s_1 s_2 \cdots s_n$ is lexicographically smaller than $t_1 t_2 \cdots t_\ell$ if

- there exists $k \leq \min(n, \ell)$ such that $s_i = t_i$ for all $1 \leq i < k$ and $s_k < t_k$
  or
- $s_i = t_i$ for all $1 \leq i \leq \min(n, \ell)$ and $n < \ell$.

Definition 3

- Uppercase letters are the uppercase English letters $(A, B, \ldots, Z)$.
- Lowercase letters are the lowercase English letters $(a, b, \ldots, z)$.

Problem A
Vowels
Time limit: 2 seconds

Alex has a string called S. Alex wants to count the number of substrings in S that contain at least one vowel. Since this is hard to do with pen and paper, you have been asked to make a computer program to help.

A letter is a vowel if it is one of the following letters: `a`, `e`, `i`, `o`, `u`.

A substring is a non-empty range of letters in a string. That is, a subsequence of consecutive letters. For example, suppose $S =$ `aba`, then the substrings are `a`, `b`, `a` (notice this is counted twice), `ab`, `ba`, and `aba`.

Input

The input consists of a single line containing a single string S. The string will only contain lowercase letters and will contain between 1 and 100 characters, inclusive.

Output

Display the number of substrings that contain at least one vowel.

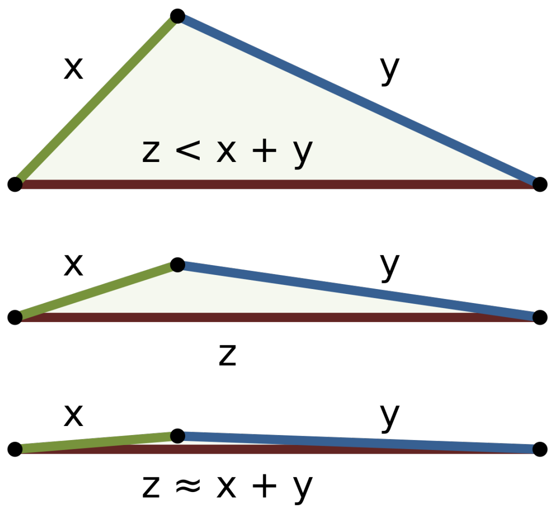
Sample Input 1	Sample Output 1
`aba`	`5`

Sample Input 2	Sample Output 2
`bcghl`	`0`

This page is intentionally left (almost) blank.

Problem B
Triangles
Time limit: 2 seconds

In mathematics, the triangle inequality states that for any triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side.



For example, suppose we have a triangle with side lengths a, b, and c. It must be the case that,

$$a \leq b + c$$
$$b \leq a + c$$
$$c \leq a + b$$

Given 3 side lengths, can you determine if they could form a triangle?

Input

The input consists of a single line containing three integers a, b, and c ($1 \leq a, b, c \leq 100$), which are the three side lengths.

Output

Display `YES` if the side lengths could form a triangle, or `NO` otherwise.

Sample Input 1	Sample Output 1
`1 2 3`	`YES`

Sample Input 2	Sample Output 2
`10 1 2`	`NO`

This page is intentionally left (almost) blank.

Problem C
Joint Venture
Time limit: 6 seconds

Liesbeth and Jan are building a robot for a course project and have discovered that they need to fit two pieces of Lego into an opening.

The opening is x centimetres wide and the sum of the lengths of the two pieces has to be precisely equal to the width of the opening, or else the robot will break during the project demonstration, with catastrophic consequences for the grades of the two students.

Luckily, Liesbeth and Jan were able to sneak into the physics laboratory late one night to measure the lengths of their remaining Lego pieces very accurately. Now they just need to select two pieces that will fit the opening perfectly.

Input

The input starts with a line containing a single integer x ($1 \leq x \leq 20$), which is the width of the opening in centimetres. This means that the width of the opening is $10\,000\,000x$ nanometres.

The second line contains a single integer n ($0 \leq n \leq 1\,000\,000$), which is the number of Lego pieces that Liesbeth and Jan have access to.

The next n lines describe the Lego pieces. Each of these lines contains a single integer ℓ ($1 \leq \ell \leq 100\,000\,000$), which is the length of the Lego piece in nanometres.

Output

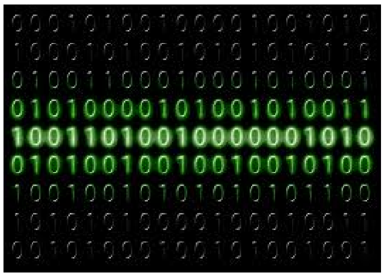
If no two pieces of Lego exist that precisely fit into the opening, display `danger`.

Otherwise, display the two lengths of the Lego pieces that fit in the opening in the form '`yes` $\ell_1 \ell_2$', with $\ell_1 \leq \ell_2$. If multiple solutions exist, display the solution that maxisises the difference between ℓ_1 and ℓ_2.

Sample Input 1	Sample Output 1
1 4 9999998 1 2 9999999	yes 1 9999999

This page is intentionally left (almost) blank.

Problem D
Bit Counting
Time limit: 2 seconds



Start with an integer, N_0, which is greater than 0. Let N_1 be the number of ones in the binary representation of N_0. So, if $N_0 = 27$ (which is 11011 in binary), then $N_1 = 4$. Similarly, if $N_0 = 5$ (which is 101 in binary), then $N_1 = 2$.

For all $i > 0$, let N_i be the number of ones in the binary representation of N_{i-1}. This sequence will always converge to one. For any starting number, N_0, let K be the minimum value of $i \geq 0$ for which $N_i = 1$. For example, if $N_0 = 31$, then $N_1 = 5$, $N_2 = 2$, $N_3 = 1$, so $K = 3$. Given a range of consecutive numbers, and a value X, how many numbers in the range have a K value equal to X?

Input

This problem contains multiple test cases.

The first line of input starts with an integer T ($1 \leq T \leq 100$), which is the number of test cases to process.

The next T lines describe the test cases. Each test case consists of a single line containing three integers L ($1 \leq L \leq 10^{18}$), H ($L \leq H \leq 10^{18}$) and X ($1 \leq X \leq 10$). L and H are the lower and upper limits of the range of integers in question and X is the target value.

Output

For each test case, display the number of integers in the range from L to H (inclusive) which have a K value equal to X.

Sample Input 1	Sample Output 1
7	1
31 31 3	0
31 31 1	0
27 31 1	3
27 31 2	0
1023 1025 0	1
1023 1025 1	1
1023 1025 2	

This page is intentionally left (almost) blank.

Problem E
Digital Roots
Time limit: 2 seconds

The *digital root* of a positive integer is found by summing the digits of the integer. If the resulting value is a single digit then that digit is the digital root. If the resulting value contains two or more digits, those digits are summed and the process is repeated. This is continued as long as necessary to obtain a single digit.

For example, consider the positive integer 24. Adding the 2 and the 4 yields a value of 6. Since 6 is a single digit, 6 is the digital root of 24. Now consider the positive integer 39. Adding the 3 and the 9 yields 12. Since 12 is not a single digit, the process must be repeated. Adding the 1 and the 2 yeilds 3, a single digit and also the digital root of 39.

Input

The input consists of a single line containing a single integer n ($1 \leq n \leq 99\,999\,999$), which is the number in question.

Output

Display the digital root of n.

Sample Input 1	Sample Output 1
`24`	`6`

Sample Input 2	Sample Output 2
`39`	`3`

This page is intentionally left (almost) blank.

Problem F
Maximum Square
Time limit: 4 seconds

Given an $N \times M$ matrix of all 1s and 0s, find the largest submatrix which is a square containing all 1s.

Input

The input starts with a line containing two integers N ($1 \leq N \leq 1\,000$), which is the number of rows, and M ($1 \leq M \leq 1\,000$), which is the number of columns.

The next N lines describe the matrix. Each of these lines contains M integers (either 0 or 1).

Output

Display the width (which is also the height) of the largest square of all 1s. If there are no 1s in the matrix, display `0`.

Sample Input 1

```
4 5
0 1 0 1 1
1 1 1 1 1
0 1 1 1 0
1 1 1 1 1
```

Sample Output 1

```
3
```

Sample Input 2

```
3 4
1 1 1 1
1 1 1 1
1 1 1 1
```

Sample Output 2

```
3
```

Sample Input 3

```
6 6
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

Sample Output 3

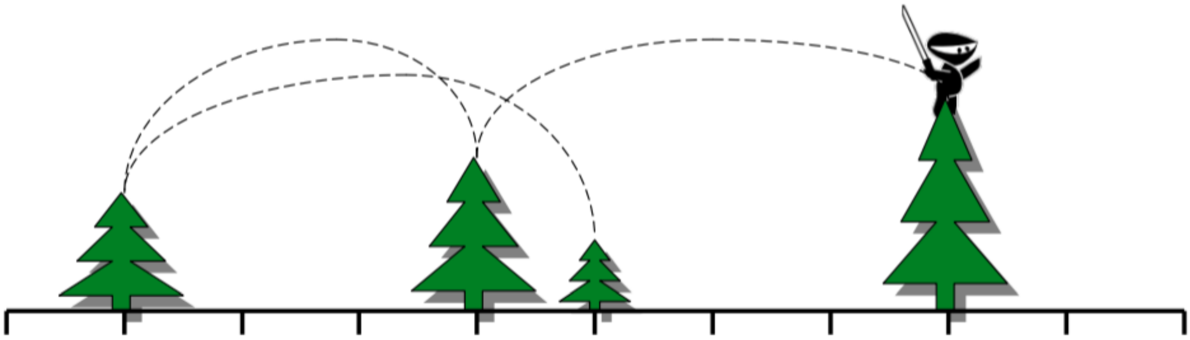
```
0
```

This page is intentionally left (almost) blank.

Problem G
The Ninja Way
Time limit: 10 seconds

As we all know, Ninjas travel by jumping from treetop to treetop. A clan of Ninjas plans to use N trees to hone their tree hopping skills. They will start at the shortest tree and make $N - 1$ jumps, with each jump taking them to a taller tree than the one they're jumping from. When finished, they will have been on every tree exactly once, traversing them in increasing order of height, and ending up on the tallest tree.

The ninjas can travel for at most a certain horizontal distance D in a single jump. To make this as much fun as possible, the Ninjas want to maximize the distance between the positions of the shortest tree and the tallest tree.



The ninjas are going to plant the trees subject to the following constraints.

- All trees are to be planted along a one-dimensional path.
- Trees must be planted at integer locations along the path, with no two trees at the same location.
- The order of the trees (from left-to-right) is pre-determined and may not be changed.
- The Ninjas can only jump so far, so every tree must be planted close enough to the next taller tree. Specifically, they must be no further than D apart on the ground (the difference in their heights doesn't matter).

Given N trees, in a specified order, each with a distinct integer height, help the ninjas figure out the maximum possible distance they can put between the shortest tree and the tallest tree, and be able to use the trees for training.

Input

This problem contains multiple test cases.

The first line of input starts with an integer T ($1 \leq T \leq 100$), which is the number of test cases to process.

Each test case starts with a line containing two integers n ($1 \leq n \leq 1\,000$), which is the number of trees, and D ($1 \leq D \leq 10^6$), which is the maximum distance that the ninjas can jump.

The next n lines describe the trees in this test case. Each of these lines contains a single integer h ($1 \leq h \leq 1\,000\,000$), which is the height of the tree. The heights of the n trees are unique within the test case.

Output

For each test case, display the maximum distance between the shortest and tallest tree. If it impossible to lay out the trees, display `-1` instead.

Sample Input 1	Sample Output 1
`` 3 4 4 20 30 10 40 5 6 20 34 54 10 15 4 2 10 20 16 13 ```	``` 3 3 -1 ```

Problem H
Palindrometer
Time limit: 3 seconds

While driving the other day, John looked down at his odometer, and it read `100000`. John was pretty excited about that. But, just one mile further, the odometer read `100001`, and John was REALLY excited! You see, John loves palindromes – things that read the same way forwards and backwards. So, given any odometer reading, what is the least number of miles John must drive before the odometer reading is a palindrome? For John, every odometer digit counts. If the odometer reading was `000121`, he wouldn't consider that a palindrome.

Input

This problem contains multiple test cases.

The first line of input starts with an integer T ($1 \leq T \leq 10\,000$), which is the number of test cases to process.

The next T lines describe the odometer readings. Each line contains a single string which contains only digits (`0`, ... , `9`) and whose length is between 2 and 9 (inclusive).

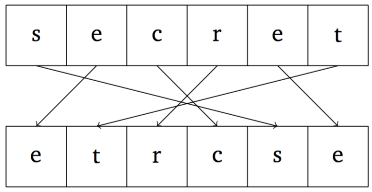
Output

For each test case, display the minimum number of miles John must drive before the odometer reading is a palindrome. This may be 0 if the number is already a palindrome.

Sample Input 1	Sample Output 1
4 100000 100001 000121 00456	1 0 979 44

This page is intentionally left (almost) blank.

Problem I
Key Insight
Time limit: 5 seconds



Alice and Bob love to send each other messages, but they don't like it when other people read their messages. Your friend Charles is very interested in what Alice and Bob send to each other, but since Alice and Bob are encrypting their messages he is unable to read them, even though he is able to intercept the encrypted messages (called "ciphertext").

Recently, Charles has not only intercepted an encrypted message, he also knows the original content of this message (which is called "plaintext"). To help him decrypt future messages between Alice and Bob, you are asked to write a program that will help him look for the encryption key.

Charles has informed you that he knows that Alice and Bob are using a transposition block cipher. This means that for each block of k characters in the message, the characters within the block are re-ordered into one of $k!$ possible permutations during encryption. Each permutation is determined by its unique corresponding encryption key. The key corresponding to the permutation shown in Figure 1 would be some representation of $(123456) \to (514362)$. Since your only task is counting (possible) keys, the actual representation is not relevant.

Fortunately, Charles does know the block size k, and he knows that the plaintext and ciphertext that he intercepted consist of one or more full blocks of length k (i.e., no incomplete blocks) that have each been encrypted with the same key. Given the plaintext M and ciphertext C that Charles has intercepted, your program will compute the number of possible encryption keys.

Input

This problem contains multiple test cases.

The first line of input starts with an integer T ($1 \leq T \leq 1\,000$), which is the number of test cases to process.

The next $3T$ lines describe the test cases. Each test case consists of 3 lines. The first line of the test case is a single integer k ($1 \leq k \leq 100$), which is the block size. The second line of the test case is a string M, which is the plaintext (M consists of only lowercase letters and is length between 1 and 100, inclusive). It is guaranteed that the length of M is a multiple of k. The third line of the test case is a string C, which is the ciphertext (C consists of only lowercase letters and is the same length as M).

Output

For each test case, display the number of possible encryption keys of size k. This number will not exceed $2^{63} - 1$. If it is impossible to obtain M from C by a transposition cipher of block size k, display `0`.

Sample Input 1	Sample Output 1
4 4 treewood ertedowo 1 nwerc ncrew 6 secret etrcse 1 impossibru youdontsay	1 0 2 0

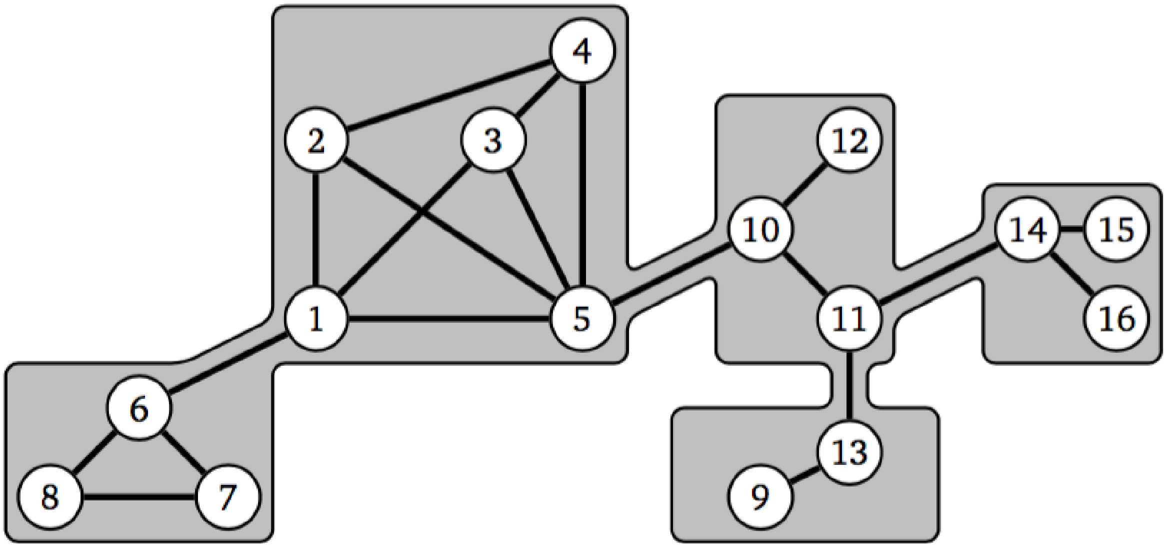
This page is intentionally left (almost) blank.

Problem J
Guards!
Time limit: 15 seconds

The royal castles in Molvania follow the design of king Sane, first of his dynasty. He ruled by divide and conquer. Therefore, all castles are built according to a hierarchical pattern based on interconnected buildings. A building consists of *halls* and *corridors* that connect halls.

Initially, a castle consists of only one building (*the main building*). When its population grows, the castle is extended as follows: A new peripheral building is constructed, attached to one of the existing buildings. Like any other building, the new building also consists of halls and corridors. An additional corridor is created to connect a hall in the existing building to a hall in the new building. That corridor is the only way to access the new building.

The number of halls in a building is at most 10.



In times of turmoil, the king monitors all corridors by strategically placing guards in halls. He asks you to determine the least number of guards required to monitor all corridors in the castle (as he wants to keep his personal guard as large as possible). Note that since the last fire, there are no doors in the castle, so we can safely assume that a guard placed in a hall can monitor all connecting corridors.

Input

This problem contains multiple test cases.

The first line of input starts with an integer T ($1 \leq T \leq 100$), which is the number of test cases to process.

Within each test case, each hall is identified by a unique number between 1 and 10000, inclusive. The input for each test case describes the castle and is recursively defined, starting with a description of the main building:

1. A line containing three integers n ($2 \leq n \leq 10$), which is the number of halls in this building, m ($1 \leq m \leq 45$), which is the number of corridors in this building, and w ($0 \leq w \leq 10$), which is the number of peripheral buildings that were later attached to this building.
2. For each of the m corridors:
   - A line containing two integers (each ≤ 10000), which are the two halls connected by this corridor. Both halls are located inside the current building.
3. For each of the w peripheral buildings:
   - A line containing two integers, describing the corridor that leads to this peripheral building. The first integer represents a hall in the current building, while the second integer represents a hall in the peripheral building.
   - The structure of the peripheral building and any newer buildings that were later attached to it, described by repeating rules 1 to 3.

The castle is fully connected: any hall is directly or indirectly reachable from any other hall. Corridors with the same start and end hall do not exist, and for every two halls there is at most one corridor between them.

Output

For each test case, display the minimum number of guards to place in halls such that all corridors in the castle are monitored.

Sample Input 1	Sample Output 1
1 5 8 2 1 2 2 4 3 4 1 3 1 5 2 5 3 5 4 5 1 6 3 3 0 6 7 7 8 8 6 5 10 3 2 2 10 11 10 12 11 13 2 1 0 13 9 11 14 3 2 0 14 15 14 16	8

Problem K
The Robot Way
Time limit: 2 seconds

You have entered a robot in a Robot Challenge. A course is set up in a $100m \times 100m$ space. Certain points are identified within the space as targets. They are ordered – there is a target 1, a target 2, etc. Your robot must start at $(0, 0)$. From there, it should go to target 1, stop for 1 second, go to target 2, stop for 1 second, and so on. It must finally end up at, and stop for a second on, $(100, 100)$.

Each target except $(0, 0)$ and $(100, 100)$ has a time penalty for missing it. So, if your robot went straight from target 1 to target 3, skipping target 2, it would incur target 2's penalty. Note that once it hits target 3, it cannot go back to target 2. It must hit the targets in order. Since your robot must stop for 1 second on each target point, it is not in danger of hitting a target accidentally too soon. For example, if target point 3 lies directly between target points 1 and 2, your robot can go straight from 1 to 2, right over 3, without stopping. Since it didn't stop, the judges will not mistakenly think that it hit target 3 too soon, so they won't assess target 2's penalty. Your final score is the amount of time (in seconds) your robot takes to reach $(100, 100)$, completing the course, plus all penalties. Smaller scores are better.

Your robot is very maneuverable, but a bit slow. It moves at 1 m/s, but can turn very quickly. During the 1 second it stops on a target point, it can easily turn to face the next target point. Thus, it can always move in a straight line between target points.

Because your robot is a bit slow, it might be advantageous to skip some targets, and incur their penalty, rather than actually maneuvering to them. Given a description of a course, determine your robot's best (lowest) possible score.

Input

The input starts with a line containing a single integer N ($1 \le N \le 1\,000$), which is the number of targets on the course.

The next N lines describe the targets. Each of these lines contain three integers x ($1 \le x \le 99$), y ($1 \le y \le 99$) and P ($1 \le P \le 100$), which denotes that this target is at (x, y) and a penalty of P is incurred if the robot misses the target. All the targets on a given course will be unique (there will be at most one target point at any location on the course).

Output

Display the smallest possible score for that course. Your answer should have an absolute or relative error of less than 10^{-6}.

Sample Input 1	Sample Output 1
1 50 50 20	143.4213562373

Sample Input 2	Sample Output 2
3 30 30 90 60 60 80 10 90 100	237.7161840722

Sample Input 3	Sample Output 3
3 30 30 90 60 60 80 10 90 10	154.4213562373

This page is intentionally left (almost) blank.