Problem A. Almost A+B?

Since a prime number has two divisors (1 and p), $p+1$ must not be divisible by 2, which is possible only when p is even. Thus, the only anisotropic prime number is 2.

Problem B. Banners Before QC

Due to the rule about `D`-jobs and `T`-jobs, in each coordinator's own order all their `D`-jobs must be completed before any of their `T`-jobs. Otherwise that coordinator would complete a `T`-job before one of their own `D`-jobs, which is forbidden.

Let S_M and S_L be the total durations of all `D`-jobs assigned to Mira and Leo. Suppose, without loss of generality, that Mira finishes her `D`-jobs not later than Leo, so $S_M \le S_L$. Then all possible violations of the global rule can only happen when Mira completes her first `T`-job: every later `T`-job of Mira and every `T`-job of Leo is completed after that moment or after Leo has already finished all his `D`-jobs.

Therefore, if Mira has at least one `T`-job, its first `T`-job has to have duration strictly greater than $S_L - S_M$. We can iterate over Mira's `T`-jobs, count how many of them can be chosen as the first one, and then order all remaining jobs arbitrarily. If Mira has no `T`-jobs, there is no extra restriction.

So the answer is the product of the factorials for the independent groups of jobs: all `D`-jobs of Mira, all `D`-jobs of Leo, all `T`-jobs of the coordinator whose `D`-jobs finish later, and all remaining `T`-jobs of the earlier coordinator. The only additional multiplier is the number of valid choices for the earlier coordinator's first `T`-job. The case $S_L < S_M$ is handled symmetrically. All computations are done modulo 998 244 353.

Problem C. Choosing Best Friend

Basic, but slow, solution is to find the maximum matching in bipartite graph between V and 2^V, where each vertex is connected with all subsets of its neighborhood, and we need to find the matching covering the first part minimizing the total weight of vertices of the second part covered by this matching.

Instead of 2^V, let us just for each vertex of the left part only keep n neighbors of minimum weight, it is enough.

Problem D. Deletion-Proof Badge

First, notice that a prime-proof code can contain only the digits 0, 1, 4, 6, 8, and 9. Indeed, if one of the digits 2, 3, 5, or 7 appeared in the code, the inspector could leave only this digit and obtain a prime number. Also, every prefix of a prime-proof code must also be prime-proof: any prime number obtainable from the prefix would be obtainable from the whole code as well. Let $d[n]$ be the number of prime-proof codes of length n.

Consider the last digit of a prime-proof code. If it is even (0, 4, 6, or 8), then this last digit cannot be part of any newly created prime number: alone it is not prime, and together with previous digits it gives a number greater than one that ends with an even digit. Therefore this case contributes $4 \cdot d[n-1]$ codes.

It remains to understand the codes whose last digit is odd (1 or 9). A brute-force search for small lengths reveals all exceptional cases, and for sufficiently large lengths only the following regular families can appear:

- `800..001`,
- `466..669`,
- codes consisting only of the digits 0, 6, and 9.

For small lengths there are a few extra prime-proof codes with an odd last digit. They are

$$91, 901, 981, 649, 949, 9801, 9081, 6049, 6649, 9469, 60049, 66049, 66649, 94669,$$

$$600049, 660049, 666049, 6000049, 6600049, 6660049.$$

Thus we set $d[1] = 5$ and, for larger n, use the recurrence

$$d[n] = 4 \cdot d[n-1] + 2 + 2 \cdot 3^{n-2} + extras[n],$$

where $extras[n]$ is the number of extra codes of length n from the list above. The term 2 corresponds to the first two regular families, and the term $2 \cdot 3^{n-2}$ counts the codes of length n over digits 0, 6, and 9 with a non-zero first digit and last digit 9.

Precompute powers of 3 and the values $d[i]$ up to the requested n, doing all operations modulo 998 244 353.

Problem E. Exactly One Of...

Note that the remainders of powers of 10 modulo 7 are periodic with period 6 (1, 3, 2, 6, 4, 5). Therefore, we find the remainder of the given number modulo 7. If it is 0, then changing a digit by $+1$ or -1 will cause the number to no longer be divisible by 7, and the answer is 0. Otherwise, we find those digit positions (modulo 6) for which adding or subtracting 1 would make the result 0 modulo 7 after the change. Then we go in steps of 6 and add 1 for each such position that is not 0 (or not 1 in the leftmost position) for subtraction, and add 1 for each such position that is not 9 for addition.

Problem F. Fix The Bad Ping

If we make a query $(1..x) \times (1..10^9)$, the result will be equal to $x \cdot 10^9 + k$, where k is the number of bad connections from exchange servers $1..x$. We can make two binary searches to find two exchange servers x_1 and x_2 with bad connections. Then in the same way we can find two fund servers y_1 and y_2 with bad connections. Now we just need to know if x_1 has bad connection with y_1 or y_2, we can find this using one more query. So in total we need $4 \log 10^9 + 1$ queries.

Problem G. Good Coach

If $n > k$, then there are not enough digits. Even $n = k$ is impossible because 0 is not a valid bus number (it should be positive). Therefore, $n < k$. Let us look at the optimal answer. If there are two numbers of lengths that differ by at least two, we can move a digit from the longer number to the shorter one, and the maximum does not increase; so we may assume that all numbers have almost equal lengths. Moreover, we know exactly that the longest number (which will be the answer) will have length $l = \lceil \frac{k}{n} \rceil$, and there will be exactly $c = k - n(l-1)$ numbers of length l. To minimize the maximum of them, firstly we need to assign the first digits to them: $1, 2, \ldots, c$. Then the number starting with c will be our answer, and, to minimize it, we need to finish it as $c, 0, c+1, c+2, \ldots, c+l-2$.

Problem H. Hi!

At p percent upload, the length of the arc is $3.6 \cdot p$ degrees. During the upload time ($d \cdot p/100$), point s has traveled a distance of $v \cdot d \cdot p/100$ degrees. Accordingly, add the length of the arc to the distance traveled and bring the coordinates into the range $[0, 360)$ (for example, in C/C++, calling the function $fmod(x, 360.0)$ brings a non-negative x into the desired range).

Problem I. Inconvenient Journey: IJ

Idea: build a graph whose vertices are pairs (x, y), where y is a platform of the station in city x. A train line between (a_i, ta_i) and (b_i, tb_i) gives an edge with cost $(0, 1)$: it does not add a shuttle trip, but it adds one train ride. Changing from one platform to another inside the same station gives cost $(1, 0)$. Then the shortest path in lexicographic order gives exactly the answer: first the number of shuttle trips, then the number of train rides.

Now we need to reduce the size of this graph. First, notice that we only need platforms that are used by at least one ICJ train line. There are at most $2m$ such platforms in total. Instead of connecting every pair

of platforms inside the same station, create one extra transfer vertex $(x, 0)$ for each city x. Add directed edges from every used platform (x, y) to $(x, 0)$ with cost $(1, 0)$, and from $(x, 0)$ back to every used platform (x, y) with cost $(0, 0)$. Passing through this transfer vertex models exactly one internal shuttle bus trip.

To start the journey, use the transfer vertex of city s as the source with distance $(0, 0)$, because Iris may choose any platform in the departure city. The answer is the minimum distance over all used platforms of city f (or, equivalently, to an extra sink connected from them with cost $(0, 0)$). The resulting graph has $O(n + m)$ vertices and edges, so Dijkstra's algorithm with lexicographic distances works fast enough.

Problem J. Just One Attempt To Hack

By calculating all the sums of the digits for Fibonacci numbers up to 2^{61} (there are 88 in total), we note that the pair of sums of digits for two consecutive numbers is unique. We construct a map $sum_1, sum_2 \to fib_1, fib_2$.

Thus, if the sum of the digits does not equal the sum of the digits of the last of the numbers, we find the sums of the digits for `fib[x]` and `fib[x+1]` and recover the original number from this pair. Otherwise, we find the sums of the digits for `fib[x-1]` and `fib[x]` and recover the original number from this pair.

Problem K. Keyword and Numeral

Read a string, output the block up to and including `=` as is, then extract two summands. According to the problem statement, there are only two possibilities: if the first character is a letter (then we also output it as is) or if the first character is a digit, in which case the summand is a number (meaning there is no need to check for the presence of letters inside). For the number, set the variable *cnt* to the number of digtis in it, and then print the numbed digit by digit, decreasing *cnt* by 1 after each digit. Every time *cnt* becomes a non-zero multiple of 3, print an apostrophe.

Problem L. Long and Random

Consider DP solution where $d[i][j]$ is the optimal answer where we take the prefix of array a of size i and remove exactly j elements from it. If you implement this solution and look at the number of removed elements in the optimal answer, you will notice that it is very close to $\frac{n}{9}$. Actually, experiments show that the optimal j with high probability is in the range $\frac{i}{9} \pm c\sqrt{i}$. So we can calculate only these values in DP, and skip the rest. The final time complexity will be $O(n\sqrt{n})$.

Problem M. Math, Nero and Seneca

With a trial division up to $B = 3\,999$ we can represent $n = p_1 p_2 \dots p_k q$ where $p_1 \le p_2 \le \dots \le p_k$ are primes less than or equal to B, and q is a positive integer all of whose prime divisors are greater than B. If $q > 1$, there is no representation at all because none of the Roman numerals is divisible by any prime divisor of q, thus the answer is 0. If $q = 1$ and $k \ge 2$ and $p_1 p_2 \le B$, then there are two different representations $n = p_1 p_2 \dots p_k = (p_1 p_2) \dots p_k$, thus the answer is 0 again. Otherwise the answer is 1: the only representation is $n = p_1 p_2 \dots p_k$ because in any other product there would be at least one composite number, but any feasible composite number is at least $p_1 p_2 > B$.