Problem A. Archery

Let's connect the targets as follows: for each x, connect target x to target 2^i if and only if the binary representation of x contains bit i. Then, it is enough to shoot at 8 targets with numbers $1, 2, 4, 8, 16, 32, 64, 128$. If the green light never lights up (otherwise the main target has already been hit at this stage), the answer is the bitwise OR of the numbers of the targets for which the yellow light lit up. The ninth shot hits the main target.

Problem B. Backing Up The Password

The problem is mainly an implementation problem for parsing an arithmetic expression.

Since there are only 10^4 possible pin code values, you can run a full search, while previously converting the formula to a polynomial form from variables a, b, c, and d (concatenation is expanded as a linear combination of variables and powers of 10, after which similarities inside brackets (if any) are reduced, brackets are expanded, and similarities are also reduced; after converting concatenation to addition, this part of the problem is reduced to the classic problem of parsing an arithmetic expression).

Problem C. Count The Repetitions

In fact the task can be solved by the Main-Lorentz algorithm: `https://cp-algorithms.com/string/main_lorentz.html`.

Problem D. Difference in Games

Note that if we can get a match in which in one of the sets the difference is 2, then the number of of solutions is infinite ($n : n - 2$ or $n - 2 : n$ depending on the winner).

Since you can win a set with a difference of at least 2, and lose — with a difference of at most 6, the worst-case difference will be +2 -6 +2 = -2. For all smaller numbers, the answer is 0.

Note that for numbers from $-2to2$, there is a possible set of matches with difference +2 -x +2 where $2 \leq x \leq 6$, any of the matches with +2 gives an infinite number of solutions.

For a difference of 3 there is a possible set of $n - 2 : n$ 6:4 6:3 matches, to obtain a difference between 4 and 10 reduce the number of lost games in won matches by the corresponding number. Thus. for the difference up to and including 10 we have an infinite number of solutions.

For a difference of 11 we have two different matches — 6:0 6:1 and 6:1 6:0 (if there is a third game, the winner of the match in it loses the game. The winner of the match loses at least 2, and in the other two wins at most 12, i.e. the difference not more than 10). For a difference of 12 we have one match — 6:0 6:0 6:0. For differences greater than 12, the answer is 0, because in one game can win no more than 6, total in two won - no more than 12, and the a lost game only decreases the point difference.

Problem E. Eligibility Test

Consider the trapezoid $ABCD$, where AB is parallel to CD, $AB \leq CD$, and consider the point O on CD. The triangles ABO, BOC and AOD must be equal, considering the equivalence for non-shared sides and that cross-angles that are equal, we have that $AB = CO = DO$, and $CD = AB \cdot 2$. Let's translate B to A, then the trapezoid degenerates into the triangle with sides $CD - AB = AB$, BC and AD.

So if one of integers is exactly twice greater, than some of remaining, and the lesser of those 2 and other 2 integers can be sides of the non-degenrate triangle, the answer is 1, otherwise the answer is 0.

Problem F. Fishing

The obvious quadratic solution: for each k and i counting the number of pairs of type (W, B), (B, W), (W, ?), (?, W), (B, ?), (?, B), (?, ?), filling the pairs with one ?, and only after they are filled, fill the pairs (?,?) with remaining fish. But the solution is too slow.

Let's create three boolean arrays, indicating the presence of the character. Consider those arrays as the polynomial of degree N. Then use FFT to find the pairwise products of the polynomials and then get the answer.

Problem G. Game Analysis

Each game increases sum $A + B$ by 1, so the total number of games is $A + B$. If A and B have non-zero fractional part, then minimal number of tied games is equal to 1, otherwise it is equal to 0 (the integer part of score P and Q can be formed by P wins of the first player and Q wins of the second player). The maximal number of the tied games is equal to the total number of games minus |A-B| (because the difference was formed by non-tied games, and one non-tied game insreases it no more than by 1).

Problem H. Hypermagic Squares

1. Let's start with a square board $(2n + 1) \times (2n + 1)$ filled with numbers from 1 to $(2n + 1)^2$ from left to right and in rows from top to bottom (standard row filling); divide the board into $n + 1$ square frames of width 1 (the center «frame» is actually a single square).

   Suppose we manage to rearrange the numbers in each box so that its four sides give the same amount, and two pairs of diagonally opposite corner cells and all pairs of orthogonally opposite side cells also give the same sum. Then these $n + 1$ frames form a magic square of the desired kind: if we delete frames sequentially, starting from the furthest, we get a series of successive magic squares, up to the middle unit square.

   Consider any of these frames, for example, with a side of $2k + 1$. Using the following algorithm, we can get a permutation of exactly the type we need.

2. Mark one corner square and the middle squares on two sides not adjacent to the selected corner square. Let $S = \{2, 4, \dots k - 2\}$.
3. If the number of marked squares is equal to the side of the frame (the same — if S is empty) — go to 7) otherwise to point (4).
4. There is exactly one pair of opposite unmarked corner squares. We mark them.
5. Choose a pair of marked corner squares that belong to the same side s. Let's choose a number i from S that was not used. We uncheck the selected pair of corner squares, then mark a pair of X and Y squares on s that are symmetrical about the middle of square s and are at a distance i from each other.
6. Jump to 3).
7. $2k + 1$ squares that are marked form some pattern. The four rotations of this template at 0, 90, 180 and 270 degrees are exactly the four sets that should be on the sides of the rearranged frame.

The numbers from the middle squares of the original frame must be transferred to the corner places, the rest is obvious.

Why does this algorithm work? Each of steps (4) by number $k - 1$ adds the same number to the sum, and steps (5) do not change the sum. Therefore, each pattern built by this algorithm covers a set of numbers with the same total sum. By choosing a different step distance (e) each time, we ensure that the four turns of the template do not have common squares that are not side averages.

Problem I. Integer With All Digits

For $N < 10$, the answer is -1.

For $N \geq 10$, the answer will look like 99...987654M_{3210}, where 9 is repeated $N - 9$ times, and M_{3210} is for the maximal integer with decimal representation consisting of four digits 3, 2, 1, 0 and that the remainder after division by 7 is equal to $7 - R$, when the R is the remainder for the integer 99...9876540000. This remainder can be calculated using the fact that 111111 is divisible by 7, so the remainder is the same as for the integer with $(N - 9) \bmod 6$ leading nines instead of $N - 9$, and then it can be calculated directly digit-by-digit. The values of M can be found by the checking all 24 permutations and keeping lexicographically biggest one for each of 7 remainders (or even on the paper).

Problem J. Joyful Points

First, note that both x and y are non-zero (else x/y or y/x is not defined).

Let $x/y = p$, $y/x = q$, p and q are integers; after multiplication we have $1 = pq$ that implies $p = q = 1$ or $p = q = -1$, so $|x| = |y|$. Both $x + y$ and $x - y$ are integers, so their sum and difference $2x$ and $2y$ are integer. So (due to $|x| = |y|$) x and y are both integer or both have the fractional part equal to 0.5. But in last case xy is not integer, so the task is reduced to counting the points with the coordinates $(\pm a, \pm a)$, where a is nonzero integer. If both a and $-a$ are between l and r, we add 4 points to the count, if only one of those values — add one point.

So we have $O(1)$ solution.

Note that the maximal answer can be equal to $4 \cdot 10^9$, so using the 64-bit signed type (long long in C++) for the variables can help avoid the overflow issues.

Problem K. King of Cabbages

Consider the following matrix A: the main diagonal elements are equal to C, on the diagonal below the main diagonal the integers $(i + 1) \cdot K[i + 1]$ are placed, other elements of the matris are equal to 0. Then the answer is $F \cdot A^t$, and the fast exponentiaton can be used (note that the elements to be recalculated are only the elements on the main diagonal and on the diagonal below). Time complexity is $O(N \log T)$.

Problem L. Letters

Each letter in the text may be either lower- or uppercase, so if the text contains N letters, the answer is $2^N - 1$. For C/C++ solutions, use the unsigned long long data type (and calculate $2^N - 1$) as the sum of 2^k to avoid overflow issues).