Reading
(Analysis)

There are two main ways to solve the problem – one of them is, sadly, too slow; however a contestant will get 50 to 60 points if implementing it well.

First solution: dynamic programming. Most of the competitors will easily see the almost trivial solution by dynamic programming. Since for a given sum N and letter L the answers depends only on the number of words with sum N – factor(L, A_i), where A_i is every other possible letter, a table dyn[26][N] is sufficient to find the answer (it also can be optimized if using iterative approach). And since for each of its cells we should check all 26 letters, the overall complexity is O(26 * 26 * N) = O(N). This solution will not get 100 points, because N can be quite large, and also the constant 26 * 26 is also of great importance for relatively small values of N.

The second solution uses a bit more non-trivial knowledge. Each competitor in this group should know that the number of paths using EXACTLY E edges in a graph, given by an adjacency matrix, is this matrix, raised to the E^{th} power. We can build a graph of the relations between letters in the alphabet and raise its matrix to the N^{th} power in order to get the answer of our problem. Although N could be quite large there is a logarithmic algorithm to do this and fit in the time limit. Here comes a problem, though. What should we do with the different distances between letters? Putting them in the matrix instead of ones and zeroes is wrong – the exponentiation of the matrix wouldn't work. Thinking in the same direction a bit more gives us the answer – since the edges are relatively short – with maximum length of only 5 – we can expand our original graph by replacing each edge with a number of new edges, each with length one, giving the same total length of the original one. Raising the new matrix to the N^{th} power will give us the number of words with complexity N. The last problem we face is that we need not only the words with complexity EXACTLY N, but also the number of words with lower complexity. This is solved by adding an artificial node with the meaning of "blank" symbol. It will have a directed edge to each of the letters and also an edge to itself. Using these edges will allow us to make words "shorter" – they will have some blanks in front of them, but this is exactly what we wanted. The extended matrix with the new node, raised to the N^{th} power will contain exactly the answer to the problem – the number of paths (words) with length (complexity) less than or equal to N.

Algorithm complexity: Since each node should be expanded into five new ones in order to guarantee paths with length up to 5, and adding one more as "empty character" we get matrix A[26*5+1][26*5+1]. The binary logarithm of 1,000,000,000 is 30, which is the number of steps we need to raise the matrix to the maximum possible power, and another K^3 algorithm for matrix multiplication, where K is the dimension of the matrix (in this case 26*5+1=131). This gives us O(131^2 * log(N)). Although a constant, 131^2 is significant enough to influence a great deal the execution time of this solution, so the contestants should not exclude it from their complexity calculations.