COP 4516 Kattis Contest Solution Sketches

Problem A: Circuit Math

URL: https://open.kattis.com/problems/circuitmath

The problem given is to evaluate a postfix expression. This algorithm is taught in COP 3502. Each time you hit a variable (letter), push the value of that variable onto the stack. Whenever you hit an operator (+, *), pop off the last two items from the stack, compute the appropriate result (or/and) and push the result back onto the stack. After processing the whole expression, the desired result is the value left on the stack.

Problem B: Hermits

URL: https://open.kattis.com/problems/hermits

For each street in the input, you just have to sum up the number of people directly on that street with the number of people on the streets that cross it. Initialize an array with the number of people already on each street, call this array total and store the number of people on each street in a separate array, street. Then, for each street crossing between street u and street v, add the number of people on street u to total[v] and similarly add the number of people on street v to the total[u]. At the end, just find the smallest count in the smallest index.

Problem C: King Arthur

URL: https://open.kattis.com/problems/kingarthur

The circumference of the table is π times the diameter. Divide this by the length of table each knight needs. Then, compare this value to the number of knights to attend the meeting. Be careful of floating point error!

Problem D: Positive Divisors

URL: https://open.kattis.com/problems/positivedivisors

The key to this problem is realizing that divisors typically come in pairs. Thus, if a is a divisor of n, then n/a is also a divisor of n. Furthermore, one of these two values must be less than or equal to the square root of n. Thus, we can stop searching for divisors once we get to the square root of n. This will allow us to run a loop a bit less than 3.2×10^7 times instead of 10^{15} times to check for divisors. So, loop up to the square root of n (to avoid floating point do `i*i <= n`), and for each divisor i, also add n/i to the list of divisors (unless it's equal to i). Then, take this list and sort!

Problem E: Positive Divisors

URL: https://open.kattis.com/problems/sidewayssorting

This one's annoying. Read in the letters into a grid and then store each column in an "object" where you store the string. Write code for a custom comparator (in Java create a class and write a compareTo method, in C++ define the < operator, etc.) that just compares the lowercase version of strings, breaking ties by the original index. Sort this list, then use the sorted case sensitive strings and output them in the grid format requested.