A. Помощь в шпионаже

Выражение для вычисления const_hash:

$$(a_N \cdot 256^{length} + a_1 \cdot 256^{length-1} + ... + a_{length} \cdot 256^0) \; mod \; 257 \; = consthash \; ,$$

где *mod 257* - это остаток от деления на 257, можно переписать в виде:

$$((a_N \cdot 256^{length}) \; mod \; 257 + (a_1 \cdot 256^{length-1}) \; mod \; 257 + ... +$$
$$+ (a_{length} \cdot 256^0) \; mod \; 257) \; mod \; 257 \; = consthash$$

Степени 256 (mod 257) можно предварительно вычислить:

$$pow_1 \; = 256^{length} \; mod \; 257$$
$$pow_2 \; = 256^{length-n-1} \; mod \; 257$$
$$pow \; = (pow_1 \; + \; pow_2) \; mod \; 257$$

Обозначим сумму:

$$S \; = (a_1 \cdot 256^{length-1}) \; mod \; 257 + ... + (a_{n-1} \cdot 256^{length-n-2}) \; mod \; 257 +$$
$$+ (a_{n+1} \cdot 256^{length-n}) \; mod \; 257 + (a_{length} \cdot 256^0) \; mod \; 257$$

тогда выражение примет вид:

$$((a_N \cdot 256^{length}) \; mod \; 257 \; + \; (a_N \cdot 256^{length-n-1}) \; mod \; 257 \; + \; S) \; mod \; 257 \; = \; consthash$$

вынесем a_N за скобку

$$(a_N \cdot pow \; mod \; 257 \; + \; S) \; mod \; 257 \; = \; consthash$$

Значение a_N можно перебрать (от 0 до 255).

B. Рисование окружностей

Основная идея этой задачи в том, чтобы проверять не попадает ли каждая точка окружности на дисплей, а наоборот для каждой точки на дисплее проверить, не является ли она точкой окружности. Далее дело техники. Проблемы могут возникнуть с переполнением (расчеты ведутся с квадратами расстояний в целых числах и типа integer уже недостаточно), с точками, когда delta_x и delta_y равны друг другу и реализовать вывод.

Память под хранение изображения дисплея не требуется, то есть решение о выводе очередной точки или пробеле можно принимать прямо во время вывода в файл очередного знака.

C. Склад

В этой задаче достаточно смоделировать работу склада. Товары можно хранить в ассоциативном массиве (например map в C++), ключ - строка (наименование товара), значение - число (количество). При операции IN прибавлять количество, при OUT - отнимать. Сложность в этой задаче может представлять считывание входных данных: нужно аккуратно реализовать разбор строки.

D. Простое счастье

Задачу можно решать рекурсивно. На каждом шаге рекурсии сначала проверяется, является ли данное число простым (в этом случае вернуть true), если нет, разделять строку на две части и запустить рекурсию с ними. Если обе части можно представить в виде простых чисел (то есть вызовы рекурсии вернули true), то значит, данное число тоже можно представить в виде простых чисел, вернуть true. Если рекурсия с номером билета вернула true, вывести YES, иначе выведите NO.
Для проверки чисел на простоту можно вычислить все простые числа, меньшие 10^6 при помощи “решета Эратосфена”.

E. Сортировка таблицы

Главная сложность задачи - реализовать ввод и вывод данных. Также для сортировки строк нужно реализовать функцию, сравнивающую строки по значению в данном столбце.

F. Матрёшки

Здесь можно реализовать простой алгоритм за $O(n^2)$. Предварительно отсортировать данные числа. Затем отбирать не уникальные - ответом будет количество итераций.
Или можно посчитать, какое количество раз встречается каждое число, максимальное количество будет ответом. Это решение за $O(n)$.

G. Очередь

Сначала нужно найти посетителя, которому “только спросить” с наименьшим номером. Если не нашлось, то ответ “!” (Василий пройдет первым). Затем нужно найти посетителя, который стоит за найденным и за которым больше никто не занимал очередь. Найти его можно например при помощи “поиска в глубину” с условием: если за каким-либо посетителем заняло очередь сразу несколько людей, то поиск нужно продолжать с посетителя с наименьшим номером.

H. Робот в столовой

Каждый посетитель берет 3 предмета, с каждым предметов робот выполняет 4 действия, то есть для каждого посетителя робот выполняет 12 действий. Ответ на задачу 12N.