A. Ферзи боятся толпы

Одним из возможных решений было идти по столбцам слева направо и ставить ферзей в строки 1 и 2, 3 и 4, ... После установки ферзя в строку n нужно установить ферзя в строку 1 и продолжить дальше.
Данное решение работает за $O(n)$ и набирает 100 баллов.

B. Юлия и делители

Для решения первых двух подзадач достаточно было для каждого запроса подсчитать количество простых чисел на отрезке между границами запроса. Также для решения второй подзадачи можно было подсчитать массив ans, где $ans[i][j]$ - количество простых чисел между i и j.

Пусть A и B - минимальная и максимальная граница всех запросов соответственно.
Создадим массив s, где $s[i] = 1$, если число $(A + i)$ является простым, иначе 0. Альтернативно, можно использовать ассоциативный массив, где где $s[i] = 1$, если число i является простым, иначе 0.
Тогда ответ на запрос (L, R) - это количество единиц в $s[L-A], s[L-A+1], \ldots, s[R-A]$. Чтобы отвечать на запрос за $O(1)$, подсчитаем на массиве s префиксные суммы.
Такое решение работает за $O((B - A) \cdot B + q)$. и набирает 100 баллов.

C. Поход

Применим подход meet-in-the-middle.
Разделим все продукты на две половины размерами $n/2$ и $n - n/2$ соответственно.
Для каждого подмножества продуктов из каждой половины подсчитаем, чему равен суммарный вес и суммарная масса. Это можно сделать перебором за $O(2^{n/2})$.

Рассмотрим задачу: дан массив объектов (w_i, v_i).
Поступают запросы (L, R): найти объект с максимальным v_i и $L \leqslant w_i \leqslant R$.
Отсортируем объекты по возрастанию w_i, при равенстве - по возрастанию v_i.
Найдём позиции l, r - позиции начала и конца подотрезка, в котором $L \leqslant w_i \leqslant R$. Это можно сделать двоичным поиском.
Теперь нужно вернуть элемент с максимальным v_i на отрезке $[l, r]$. Для этого можно использовать разреженную таблицу или другую подходящую структуру данных.
Таким образом, для массива размера n и q запросов данная задача решается за
$O(T(sort) + T(build\_sparse\_table) + q \cdot (2 \cdot T(binary\_search) + T(get\_max)) =$
$O(n \cdot \log(n) + n \cdot \log(n) + q \cdot (\log(n) + 1)) = O((n + q) \cdot \log(n))$.

Переберём подмножество взятых продуктов из первой половины, пусть оно кодируется битовой маской $mask_1$. Для него нужно найти подмножество взятых продуктов из второй половины с максимальной стоимостью и весом, лежащим в интервале $[L - total\_weight(mask_1), R - total\_weight(mask_1)]$ (это в точности описанная выше задача).
Таким образом, решение всей задачи работает за $O(2^{n/2} \cdot \log(n))$ и набирает 100 баллов.

D. Глеб и печеньки

Будем поддерживать позицию последней встреченной буквы 'b', назовём её $last$.
Подсчитаем и запомним в массив $dist$ количество букв между первой после $last$ буквой 'B' и ближайшей справа от неё буквой 'b'. Обновим $last$. Это можно сделать за линейное время.
После чего нужно из массива $dist$ брать минимальный элемент, пока сумма невзятых элементов не превышает k. Для этого нужно отсортировать массив $dist$.
Такое решение работает за $O(n \cdot \log(n))$, где $n = |s|$, и набирает 100 баллов.

E. Бизнес-гусеницы

Воспользуемся динамическим программированием.
Пусть $dp[i] = true$, если осталось i камней и выигрывает та гусеница, которая ходит сейчас, иначе $dp[i] = false$.
Если осталось i камней, то можно выиграть, если перевести соперника в проигрышное состояние за один ход.
Таким образом, $dp[i] = dp[i-1]$ or $dp[i-2]$ or $dp[i-6]$.
Такое решение работает за $O(n)$ и набирает 50 баллов.

Рассмотрим, как выглядит последовательность dp, начиная с нулевого элемента.
$0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, ...$ $(0 - false, 1 - true)$
Несложно понять, что у этой последовательности период 7, поэтому ответ для чисел с одинаковым остатком по модулю 7 будет равным.
Такое решение работает за $O(1)$ и набирает 100 баллов.