Общие требования

В этом задании еще нельзя пользоваться готовыми бинарными деревьями, например, деревом поиска или кучей (`std::set`, `std::priority_queue` на C++); если они вам понадобились, надо реализовать их самостоятельно.
Начиная с этого задания можно пользоваться стандартными алгоритмами сортировки и другими (`std::sort` и остальная библиотека `<algorithm>` на C++), их больше не надо писать самостоятельно.

Задача 3-1. Известно, что каждая клетка организма человека содержит ДНК, которая представляет из себя цепочку нуклеотидов, кодируемых символами A, G, T, C, и, кроме того, что эти цепочки у близких родственников должны быть похожими.

Для определения того, принадлежат ли два заданных ДНК s_1 и s_2 близким родственникам, вам нужно решить следующую задачу. Необходимо найти такую позицию i в строке s_1, что, если приложить s_2 к s_1, начиная с позиции i, то, во-первых, каждому символу s_2 будет соответствовать какой-то символ s_1 (то есть s_2 не выйдет за пределы s_1), а во-вторых, количество соответствующих друг другу совпадающих символов в s_1 и s_2 будет максимально возможным. Другими словами, $i + length(s_2) \le length(s_1)$ и мощность множества $\{j : s_2[j] = s_1[i + j]\}$ — максимально возможная из всех таких i. Если таких позиций i несколько, найдите первую из них (с наименьшим номером).

На входе две непустые строки s_1 и s_2 длины не более 200 000, причем s_2 не длиннее, чем s_1. Выведите номер искомой позиции. Позиции нумеруются с единицы.

Пример входа	Пример выхода
AGTCAGTC GTC	2
AAGGTTCC TCAA	5

Задача 3-2. На дороге в некоторых местах разбросаны золотые монеты. Для каждой монеты известно ее местоположение, которое задается одним целым числом — расстоянием в метрах от начальной отметки. Все монеты расположены правее начальной отметки. Али-баба бегает по дороге и собирает монеты, начиная делать это в момент времени 0. За одну секунду он пробегает ровно один метр. У каждой монеты есть крайний срок, до которого (включительно) ее нужно подобрать, иначе монета исчезнет. Али-баба должен собрать все монеты и сделать это за минимально возможное время. Он может стартовать в любой точке прямой, собирать монеты в произвольном порядке, но обязательно нужно успеть собрать все монеты и при этом минимизировать затраченное время.

В первой строке входа задано число n — количество монет ($1 \le n \le 1000$). В каждой из следующих строк — по 2 целых числа, первое из которых задает положение монеты, а второе — крайний срок в секундах, за который нужно успеть собрать эту монету. Координаты монеток от 0 до 1 000 000. Ограничения по времени также не превосходят 1 000 000. Выведите одно число — минимальное время, за которое Али-баба может собрать все монеты. Если Али-баба не может успеть собрать все монеты, выведите строку No solution.

Пример входа	Пример выхода
5 1 3 3 1 5 8 8 19 10 15	11
5 1 5 2 1 3 4 4 2 5 3	No solution

Задача 3-3. Пете поручили написать менеджер памяти для новой стандартной библиотеки языка H++. В распоряжении у менеджера находится массив из N последовательных ячеек памяти, пронумерованных от 1 до N. Задача менеджера — обрабатывать запросы приложений на выделение и освобождение памяти. Запрос на выделение памяти имеет один параметр K. Такой запрос означает, что приложение просит выделить ему K последовательных ячеек памяти. Если в распоряжении менеджера есть хотя бы один свободный блок из K последовательных ячеек, то он обязан в ответ на запрос выделить такой блок. При этом наш менеджер выделяет память из самого длинного свободного блока, а если таких несколько, то из них он выбирает тот, у которого номер первой ячейки — наименьший. После этого выделенные ячейки становятся занятыми и не могут быть использованы для выделения памяти, пока не будут освобождены. Если блока из K последовательных свободных ячеек нет, то запрос отклоняется. Запрос на освобождение памяти имеет один параметр T. Такой запрос означает, что менеджер должен освободить память, выделенную ранее при обработке запроса с порядковым номером T. Запросы нумеруются, начиная с единицы. Гарантируется, что запрос с номером T — запрос на выделение, причем к нему еще не применялось освобождение памяти. Освобожденные ячейки могут снова быть использованы для выделения памяти. Если запрос с номером T был отклонен, то текущий запрос на освобождение памяти игнорируется. Требуется написать симуляцию менеджера памяти, удовлетворяющую приведенным критериям.

В первой строке входа два числа N и M — количество ячеек памяти и запросов соответственно ($1 \le N \le 2^{31} - 1$, $1 \le M \le 10^5$). Каждая из следующих M строк содержит по одному числу. $(i + 1)$-я строка содержит положительное число K, если i-й запрос — запрос на выделение K ячеек памяти ($1 \le K \le N$), и отрицательное число $-T$, если i-й запрос — запрос на освобождение памяти, выделенной по запросу номер T ($1 \le T < i$).

Для каждого запроса на выделение памяти выведите в выход одно число на отдельной строке с результатом выполнения этого запроса. Если память была выделена, выведите номер первой ячейки памяти в выделенном блоке, иначе выведите число -1.

Пример входа	Пример выхода
6 8 2 3 -1 3 3 -5 2 2	1 3 -1 -1 1 -1

Задача 3-4. (Задача о скользящей k-й статистике.) В первой строке входа — три целых числа n, m, k. Во второй строке n целых чисел, задающих массив чисел, по которому будут двигаться два указателя, l и r. Изначально оба указателя направлены на самый первый элемент массива. Есть две операции: L — сдвинуть указатель l на один элемент вправо и R — сдвинуть указатель r на один элемент вправо. В третьей строке входного файла — m символов R или L, без пробелов, в одну строку. Это порядок выполняемых операций. Гарантируется, что указатель l никогда не "обгоняет" указатель r. Гарантируется, что указатель r никогда не выйдет за пределы массива. При этом $1 \le n, k \le 100000$, $0 \le m \le 2n - 2$. Все числа в массиве неотрицательные и не превосходят 10^9.

Выведите ровно m строк, в каждой — ровно по одному целому числу. После выполнения каждой из операций нужно вывести k-е в порядке возрастания число среди всех чисел от l до r включительно, либо -1, если всего чисел от l до r меньше, чем k.

Пример входа	Пример выхода
7 4 2 4 2 1 3 6 5 7 RRLL	4 2 2 -1
4 6 1 1 2 3 4 RLRLRL	1 2 2 3 3 4