Задача A. Компаратор

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Юный программист Саша написал свою первую тестирующую систему. Он так обрадовался тому, что она скомпилировалась, что решил пригласить школьных друзей на свой собственный контест.

Но в конце тура выяснилось, что система не умеет сортировать команды в таблице результатов. Помогите Саше реализовать эту сортировку.

Команды упорядочиваются по правилам ACM:

1. по количеству решённых задач в порядке убывания;
2. при равенстве количества решённых задач – по штрафному времени в порядке возрастания;
3. при прочих равных — по номеру команды в порядке возрастания.

Используйте *std::vector* в этой задаче и собственный компаратор.

Формат входных данных

Первая строка содержит натуральное число n ($1 \leqslant n \leqslant 10^5$) — количество команд, участвующих в контесте.

В i-й из следующих n строк записано количество решенных задач S ($0 \leqslant S \leqslant 100$) и штрафное время T ($0 \leqslant T \leqslant 10^5$) команды с номером i.

Формат выходных данных

В выходной файл выведите n чисел — номера команд в отсортированном порядке.

Пример

стандартный ввод	стандартный вывод
5 3 50 5 720 1 7 0 0 8 500	5 2 1 3 4

Задача B. Зверюшки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Уже долгое время в Институте Мутантов, Фич Технологических и Иностранных языков разводят милых разноцветных зверюшек. Для удобства каждый цвет обозначен своим номером, всего цветов не более 10^9.

В один из прекрасных дней в питомнике случилось чудо: все зверюшки выстроились в ряд в порядке возрастания цветов. Пользуясь случаем, лаборанты решили посчитать, сколько зверюшек разных цветов живет в питомнике, и, по закону жанра, попросили вас написать программу, которая поможет им в решении этой нелегкой задачи.

Используйте для решения этой задачи готовые функции *STL* для *std::vector* и библиотеку *algorithm*.

Формат входных данных

В первой строке входного файла содержится единственное число N ($0 \leqslant N \leqslant 10^5$) — количество зверюшек в Институте.

В следующей строке находятся N упорядоченных по неубыванию неотрицательных целых чисел, не превосходящих 10^9 и разделенных пробелами — их цвета.

В третьей строке файла записано число M ($1 \leqslant M \leqslant 100000$) — количество запросов вашей программе, в следующей строке через пробел записаны M целых неотрицательных чисел (не превышающих $10^9 + 1$).

Формат выходных данных

Выходной файл должен содержать M строчек. Для каждого запроса выведите число зверюшек заданного цвета в питомнике.

Пример

стандартный ввод	стандартный вывод
10 1 1 3 3 5 7 9 18 18 57 5 57 3 9 1 179	1 2 1 2 0

Задача C. Встреча

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Напишите программу, которая определяет, сколько раз встречается заданное число x в данном массиве. Используйте последовательный контейнер для решения задачи.

Формат входных данных

В первой строке задается одно натуральное число N, не превосходящее 1000 — размер массива.

Во второй строке вводятся N чисел — элементы массива (целые числа, не превосходящие по модулю 1000).

В третьей строке содержится одно целое число x , не превосходящее по модулю 1000.

Формат выходных данных

Вывести одно число — сколько раз встречается x в данном массиве.

Пример

стандартный ввод	стандартный вывод
5 1 2 3 4 5 3	1

Задача D. Список смежности

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

В этой задаче необходимо организовать неориентированный граф, на котором поддерживаются следующие операции:

1. $AddEdge(u, v)$ — добавить в граф ребро между вершинами (u, v).
2. $Vertex(u)$ — вывести список вершин, смежных с вершиной u.

Петель и кратных ребер в графе нет. Для решения данной задачи используйте последовательные контейнеры из STL.

Формат входных данных

В первой строке входного файла содержится целое число N ($1 \leqslant N \leqslant 10^6$) — количество вершин в графе.

В следующей строке находится целое числоK ($0 \leqslant K \leqslant 10^6$) — число операций, затем идет описание операций — каждое в своей строке.

Операции имеют следующий формат: «1 u v » или «2 u », обозначающие соответственно операции $AddEdge(u, v)$ и $Vertex(u)$.

Гарантируется, что суммарное количество чисел, которое будет необходимо вывести при выполнении всех операций Vertex не превосходит $2 \cdot 10^5$.

Формат выходных данных

В выходной файл для каждой команды Vertex необходимо на отдельной строке вывести список смежных вершин указанной вершины. Вершины списка смежности нужно выводить в порядке добавления соответствующих ребер в граф.

Пример

стандартный ввод	стандартный вывод
4 4 1 1 2 1 2 3 2 2 1 1 3	1 3

Замечание

Подумайте, как реализовать команды выше в случае ориентированного графа или если в нем есть петли.

Подумайте, как реализовать хранение графа с кратными ребрами.

Задача E. Множество строк

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Реализуйте структуру данных «множество строк». Хранимые строки — непустые последовательности длиной не более 10 строчных латинских букв. Структура данных должна поддерживать операции:

1. Добавление строки в множество;
2. Удаления строки из множества;
3. Проверки принадлежности данной строки множеству.

Максимальное количество элементов в хранимом множестве не превосходит 10^6.
Для решения задачи используйте *std::unordered_ set*.

Формат входных данных

Каждая строка входных данных задает одну операцию над множеством. Запись операции состоит из типа операции и следующей за ним через пробел строки, над которой проводится операция. Тип операции — один из трех символов:

- + означает добавление данной строки в множество;
- − означает удаление строки из множества;
- ? означает проверку принадлежности данной строки множеству.

Общее количество операций во входном файле не превосходит 10^6. Список операций завершается строкой, в которой записан один символ #.
При добавлении элемента в множество не гарантируется, что он отсутствует в этом множестве.
При удалении элемента из множества не гарантируется, что он присутствует в этом множестве.

Формат выходных данных

Программа должна вывести для каждой операции типа ? одну из двух строк «YES» или «NO», в зависимости от того, встречается ли данное слово в множестве.

Пример

стандартный ввод	стандартный вывод
+ hello + bye ? bye - bye ? bye ? hello #	YES NO YES

Замечание

Подумайте, почему бы не использовать *std::set*.
В случае TL почитайте про быстрый ввод и вывод в C++ (с обычным вводом решение работает за $1,9$ секунд): https://codeforces.com/blog/entry/5217?locale=ru

Задача F. По верхней границе

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Реализуйте структуру данных, которая поддерживает множество S целых чисел, с котором разрешается производить следующие операции:

- $add(i)$ — добавить в множество S число i (если он там уже есть, то множество не меняется);
- $next(i)$ — вывести минимальный элемент множества, не меньший i. Если искомый элемент в структуре отсутствует, необходимо вывести -1.

Для решения задачи используйте *std::set* и его методы.

Формат входных данных

Исходно множество S пусто. Первая строка входного файла содержит число n — количество операций ($1 < n < 3 \cdot 10^5$).

Следующие n строк содержат операции. Каждая операция имеет вид:

- $+\ i$ — $add(i)$
- $?\ i$ — $next(i)$

Если операция + идет во входном файле в начале или после другой операции +, то она задает операцию $add(i)$. Если же она идет после запроса ?, и результат этого запроса был y, то выполняется операция $add((i + y)\ mod\ 10^9)$.

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до 10^9.

Формат выходных данных

Для каждого запроса выведите одно число — ответ на запрос.

Пример

стандартный ввод	стандартный вывод
6 + 1 + 3 + 3 ? 2 + 1 ? 4	3 4

Задача G. Словарь

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

Вам дан словарь, состоящий из пар слов. Каждое слово является синонимом к парному ему слову. Все слова в словаре различны. Для каждого данного слова определите его синоним.

Для решения данной задачи используйте *std::unordered_map*

Формат входных данных

Программа получает на вход количество пар синонимов N ($0 \leqslant N \leqslant 10^5$). Далее следует N строк, каждая строка содержит ровно два слова-синонима.

Затем идет число Q ($1 \leqslant Q \leqslant 10^5$) — количество запросов к словарю. Далее на каждой следующей из Q строк идет слово, к которому надо вывести синоним.

Формат выходных данных

Программа должна вывести синонимы к данным слову на отдельных строках.

Пример

стандартный ввод	стандартный вывод
3 car plane mouse cat base stream 3 plane stream base	car base stream

Замечание

Подумайте, почему бы не использовать *std::map*. Используйте быстрый ввод.

Задача N. Банк

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

У банка есть клиенты. Каждый клиент имеет ровно один счет.

Напишите программу, которая будет выполнять последовательность запросов таких двух видов:

1. Начинается с числа 1, затем через пробел следует имя клиента (слово из латинских букв), далее через пробел идет сумма денег, которую клиент кладет или берет из счета в банке (целое число, не превышает по модулю 10000).

2. Начинается с числа 2, через пробел следует имя клиента. На каждый такой запрос программа должна ответить какая сумма в данный момент есть на счету заданного клиента. Если такое имя клиента пока ни разу не упоминалось в запросах вида 1, выводите вместо числа слово «ERROR».

В начале работы программы у всех клиентов на счету 0. Затем суммы могут становиться как положительными, так и отрицательными.

Обратите внимание, что в ситуации, когда клиент снял суммарно ровно столько же денег, сколько положил, сумма на счете становится равной 0, но, раз его имя уже встречалось, нулевое значение не является основанием выводить «ERROR».

Для решения задачи используйте ассоциативный контейнер.

Формат входных данных

Первая строка стандартного входного потока количество запросов N ($1 \leqslant N \leqslant 10^5$). Далее следуют N строк в каждой из которых описан один из двух описанных выше видов запроса.

Формат выходных данных

На каждый запрос 2-го вида нужно вывести текущее значение на счету заданного клиента (или слово «ERROR»).

Пример

стандартный ввод	стандартный вывод
7 1 asdf 3 1 zxcv 5 2 asdf 1 asdf -2 2 asdf 2 lalala 2 zxcv	3 1 ERROR 5

Задача I. Частотность

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дан текст. Выведите все слова, встречающиеся в тексте, по одному на каждую строку. Слова должны быть отсортированы по убыванию их количества появления в тексте, а при одинаковой частоте появления в лексикографическом порядке.

Формат входных данных

Вводится текст — последовательность строк через пробел или перенос строки.

Формат выходных данных

Выведите ответ на задачу.

Примеры

стандартный ввод	стандартный вывод
hi hi what is your name my name is bond james bond my name is damme van damme claude van damme jean claude van damme	damme is name van bond claude hi my james jean what your
oh you touch my tralala mmm my ding ding dong	ding my dong mmm oh touch tralala you

Замечание

Указание. После того, как вы создадите словарь всех слов, вам захочется отсортировать его по частоте встречаемости слова. Желаемого можно добиться, если создать список, элементами которого будут пары из двух элементов: частота встречаемости слова и само слово. Например, $[(2, hi), (1, what), (3, is)]$. Тогда стандартная сортировка будет сортировать список пар, при этом кортежи сравниваются по первому элементу, а если они равны, то по второму. Это почти то, что требуется в задаче.

Задача J. База данных

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дана база данных о продажах некоторого интернет-магазина. Каждая строка входного файла представляет собой запись вида *покупатель-товар-количество* , где *покупатель* — имя покупателя (строка без пробелов), *товар* — название товарас (строка без пробелов), количество — количество приобретённых единиц товара.

Создайте список всех покупателей и для каждого покупателя подсчитайте количество приобретенных им единиц каждого вида товаров.

Формат входных данных

Во входном файле записано не более 10^5 строк в указанном формате.

Имена покупателей и названия товаров представляют собой строки из заглавных и строчных латинских букв не длиннее 10 символов. Количество товара в каждой покупке — натуральное число, не превышающее 10^6.

Формат выходных данных

Выведите список всех покупателей в лексикографическом порядке, после имени каждого покупателя выведите двоеточие, затем выведите список названий всех приобретенных данным покупателем товаров в лексикографическом порядке, после названия каждого товара выведите количество единиц товара, приобретенных данным покупателем.

Информация о каждом товаре выводится в отдельной строке.

Пример

стандартный ввод	стандартный вывод
Ivanov paper 10 Petrov pen 5 Ivanov marker 3 Ivanov paper 7 Petrov envelope 20 Ivanov envelope 5	Ivanov: envelope 5 marker 3 paper 17 Petrov: envelope 20 pen 5

Задача K. Недешевый калькулятор

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Студенты ФПМИ придумали новую версию калькулятора. Этот калькулятор берет с пользователя валюту за совершаемые операции. Стоимость каждой операции в *ФПМИ-коинах* равна 5% от числа, которое является результатом операции.

На этом калькуляторе требуется вычислить сумму N натуральных чисел (числа, так уж и быть, известны). Нетрудно заметить, что от того, в каком порядке мы будем складывать эти числа, иногда зависит, в какую *ФПМИ-коинов* нам обойдется вычисление суммы чисел (то есть оказывается нарушено великое правило «от перестановки мест слагаемых сумма не меняется»).

Например, пусть нам нужно сложить числа $10, 11, 12, 13$. Подумайте, как от порядка вычислений зависит ответ. Напишите программу, используя *std::priority_queue*, которая будет определять, за какую минимальную сумму *ФПМИ-коинов* можно найти сумму данных N чисел.

Формат входных данных

Во входном файле записано число N ($2 \leqslant N \leqslant 10^5$). Далее идет N натуральных чисел, которые нужно сложить, каждое из них не превышает 10^4.

Формат выходных данных

В выходной файл выведите, сколько *ФПМИ-коинов* нам потребуется на нахождение суммы этих N чисел.

Пример

стандартный ввод	стандартный вывод
4 10 11 12 13	4.6

Замечание

Подумайте, какой еще контейнер *STL* мог бы решить эту задачу, почему рекомендована именно приоритетная очередь.

Задача L. Беды с английским

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вы на грани вылета из института.

И неудивительно; на английский вы не ходили, по курсу алгоритмов вы не захотели получать трояк и пошли на пересдачу, на которую не пришли, так как вместо этого пошли с другом гамать. Еще вы не ходили на физру, а впереди матан, матлог и экзамен по ООП.

Возможно, к кафедре иностранных языков вы найдете подход. Там есть активно посещаемый сайт, цель которого — выкладывать очередное огромное домашнее задание. И сайт этот в последнее время жутко тормозит, так как слишком много пользователей пишут тесты онлайн. Кафедра вам сказала, что если вы «ускорите работу сайта», то, возможно, вам пойдут навстречу.

После некоторого изучения кода вы поняли, что слишком часто происходят обращения к базе данных, а именно, к таблице пользователей. Смысла в этом особого нет, поскольку набор посетителей сайта в каждый момент времени ограничен, и профили пользователей можно для быстрого доступа хранить в памяти, пока они осуществляют навигацию по сайту.

Нужно просто написать кэширующую версию функции *get_user(int user_id)*, которая возвращает профили пользователей, хранящиеся в базе данных.

Логику кэширования Вы придумали следующую. Пусть размер кэша равен M. Тогда функция с кэшированием значений должна хранить список закэшированных профилей пользователей (будем профили пользователей называть просто пользователями). Размер списка не более M. Он конструируется по следующему принципу.

Когда функция вызывается с некоторым аргументом *user_id* необходимо проверить, есть ли пользователь с таким идентификатором в списке, и если есть, вернуть закэшированный профиль, предварительно переместив его в самое начало списка. Если же такого пользователя нет, то нужно обратиться к базе данных и поместить пользователя в начало списка. Если размер списка стал больше M (а именно, $M + 1$), необходимо удалить из него последний элемент.

Таким образом, в списке всегда будут храниться закэшированные ответы для последних M использованных уникальных значений аргумента функции. Вы хотите изучить, как меняется польза от кэширования в зависимости от значения M на последовательности реальных данных. Для решения этой задачи используйте ассоциативный контейнер и очередь с приоритетами.

Формат входных данных

Первая строка входа содержит целое число M ($0 < M < 5 \cdot 10^5$) — размер кэша. Затем идёт последовательность целых чисел из диапазона $[0, 2 \cdot 10^7]$, разделённых пробельными символами. Количество чисел не больше $5 \cdot 10^5$.

Формат выходных данных

Для каждого введённого числа необходимо вывести 0, если использовалось закэшированное значение, а иначе 1. Другими словами, для каждого введенного числа нужно выводить 0, если среди последних M уникальных чисел его нет, а иначе вывести 0. Цифры разделяйте пробелом.

Примеры

стандартный ввод	стандартный вывод
1 1 2 3 4 5 6	1 1 1 1 1 1
4 7 7 7 7 7	1 0 0 0 0
2 1 2 3 4 5 5 1 1 2 2 3 3 4 4 5 5 6 6 7	1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1

Задача M. Радио

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Компания «Аудио Квадрат» захватила значительную долю рынка музыкальных товаров и услуг. В частности она делает персональное радио для каждого человека, которое учитывает его музыкальные предпочтения и текущее настроение.

Но параллельно, just for fun, компания запускает проект *общественного радио*. Это радио, в котором песня выбирается в соответствии с пожеланиями множества проголосовавших пользователей.

Алгоритм общественного радио безумно прост. Посетители интернет сайта «Аудио Квадрата» постоянно голосуют за отдельные композиции (с одного IP можно проголосовать в течении 10 минут только за одну композицию). Голос может иметь вес *score*. Для каждой композиции считаются очки — *track_score*. Следующей композицией, которая будет играть на радио, станет та, которая имеет максимальное количество очков. Если таких композиций несколько, то должна играть та, которая имеет минимальный идентификатор. В момент, когда композиция запускается на радио, её *track_score* становится равным -1.

Чтобы исключить попытки накручивания счётчика ределёнными группами лиц, компания решила принимать голоса с одного IP не чаще раза в 10 минут (два голоса могут быть приняты, если между ними не менее шестисот секунд).

Вы являетесь главным разработчиком компании «Аудио Квадрат». И хотя у вас в подчинении множество высококлассных программистов, вы решили вспомнить старые добрые времена и запрограммировать этот алгоритм сами.

Общественных радио будет много, поэтому важно, чтобы каждое из них работало достаточно эффективно, то есть не отнимало много процессорных ресурсов и оперативной памяти.

Необходимо написать программу, реализующую алгоритм работы одного общественного радио. Взаимодействие с программой будет осуществляться через стандартный поток ввода/вывода согласно заданному ниже протоколу.

Для решения этой задачи используйте ассоциативные контейнеры и очереди с приоритетами.

Формат входных данных

Каждая строчка входа — это определенная команда. Всего 3 типа команд:

- $VOTE$ ip track_id score time
- GET
- $EXIT$

Команда $VOTE$ меняет число очков *track_score* определённой композиции. Она получает четыре аргумента:

- *ip* — IP адрес компьютера, с которого пришел голос; четыре целых числа из промежутка $[0, 255]$, разделённых точкой
- *track_id* — численный идентификатор композиции; натуральное число из отрезка $[1, 2 \cdot 10^7]$;
- *score* — количество очков, которое нужно добавить к текущим очкам композиции *track_score* — целое число из отрезка $[-100, 100]$;
- *time* — момент времени в секундах от некоторого фиксированного момента времени — целое число из промежутка $[0, 2 \cdot 10^9]$.

На команду $VOTE$ нужно отвечать новым значением очков музыкальной композиции *track_id* (даже если эти очки не изменились).

Команда GET используется для получения следующей композиции. Ваша программа должна отвечать на неё парой $track\_id$ и $track\_score$ (вывести эти два числа в одной строке, разделив их пробелом). Сразу после выполнения этой команды новое значение $track\_score$ для этой композиции должно стать равным -1.

Команда $EXIT$ находится в последней строке входа. При её получении необходимо вывести строку «OK». Число команд на входе не более 100001. Изначально значение $track\_score$ для всех композиций равно 0.

Команды во входных данных упорядочены по параметру $time$ (для двух из трёх команд этот параметр просто не передаётся, так как не нужен).

Формат выходных данных

Выход должен содержать ровно столько же строчек, что и вход. Каждая строчка выхода — это ответ на соответствующую команду из входа.

Примеры

стандартный ввод	стандартный вывод
GET VOTE 1.1.1.1 1 -1 1 VOTE 1.1.1.1 2 1 2 VOTE 1.1.1.1 1 2 4 VOTE 1.1.1.1 1 4 20045 GET GET GET VOTE 194.85.81.128 3 -1 20049 EXIT	1 0 -2 0 -2 2 1 2 2 0 3 0 -2 OK
GET GET GET VOTE 192.168.0.1 2 2 11111111 EXIT	1 0 2 0 3 0 1 OK

Задача N. Текстовый редактор

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Реализовать функцию Undo-Redo для текстового редактора. Программа получает на входе последовательность символов, вводимую пользователем. Для разделения строк используется символ n. В тексте используются следующие спецсимволы:

- < — переход на один символ влево или в конец предыдущей строки
- > — переход на один символ вправо или в начало следующей строки
- ^— переход на строку выше
- | — переход на строку ниже
- # — удаление символа (нажатие Del)
- @ — отмена последнего действия (Undo)
- $ — повтор последнего отмененного действия (Redo)

Undo и Redo могут встречаться подряд несколько раз. Программа должна вывести текст, получившийся в результате такого редактирования.

Вот правила, более чётко определяющие поведение текстового редактора. Каждая строка текста заканчивается символом перевода строки, в том числе последняя. Вначале текст содержит одну пустую строку. Текущее место редактирования опредляется позицией в тексте, которую назовём курсором. Курсор никогда не выходит за пределы текста: например, нажатие "вверх"во время редактирования первой строки не приводит к перемещению курсора. Если курсор с помощью символов ^(вверх) или | (вниз) перемещается на строку, длина которой меньше его текущей горизонтальной позиции, то курсор ставится в конец строки. Отменить/повторить можно действия добавления и удаления символа (в том числе перевода строки, в результате чего строка расщепляется или сливается с соседней). Нерезультативное действие не запоминается (например нажатие Del, когда курсор находится в самом конце текста). Команда Undo в момент, когда нет действий для отмены, не является ошибкой, также как и команда Redo, если нет действий для повтора. После отмены действия курсор возвращается в позицию, которую имел непосредственно перед выполнением этого действия.

Формат входных данных

На вход подаётся последовательность латинский букв, цифр, пробелов, знаков препинания и переводов строк, а также описанных выше специальных символов — суммарной длиной не более 500000. Гарантируется, что при редактировании текста никакая строка не становилась длиннее 100 символов.

Формат выходных данных

Вывести получившийся текст.

Пример

стандартный ввод	стандартный вывод
I a<#often set and wish that i Could be a kite up in the sky,^<#I|>>>>>>Co And ridee@ upon the@@$$$$$$ breeze and go Whichever way I CHA@@@chanced to blow.	I often set and wish that I Could be a kite up in the sky, And ride upon the breeze and go Whichever way I chanced to blow.