Задача A. Плохая подстрока

Имя входного файла:	`badsubs.in`
Имя выходного файла:	`badsubs.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Найдите, сколько существует строк заданной длины n, состоящих только из символов «`a`», «`b`» и «`c`», но не содержащих подстроки «`ab`».

Формат входных данных

Во входном файле задано n ($0 \leqslant n \leqslant 22$).

Формат выходных данных

Выведите количество таких строк.

Примеры

badsubs.in	badsubs.out
0	1
3	21
11	46368

Задача B. Сообщение об ошибке

Имя входного файла:	`errmess.in`
Имя выходного файла:	`errmess.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Правильные скобочно-палочные последовательности (ПСПП) определяются рекурсивно следующим образом:

1. Пустая строка является ПСПП.
2. Если A и B — две ПСПП, то AB (строка, полученная приписыванием строки B справа к строке A) — тоже ПСПП.
3. Если A и B — две ПСПП, то строка `(`A`|`B`)` также является ПСПП.

Например, «`(|)`» является ПСПП, так как получается по пункту 3 определения для пустых строк A и B, которые являются ПСПП по пункту 1. Строка «`((|)|)(|)`» также является ПСПП: по пункту 2 её можно составить из строк A = `((|)|)` и B = `(|)`. В свою очередь, строка «`((|)|)`» является ПСПП по пункту 3 определения для A = `(|)` и пустой строки B.

Задана строка, состоящая из символов «`(`», «`|`» и «`)`». Если эта строка является ПСПП, выведите сообщение «`correct, length =` x», где x — длина строки. В противном случае выведите сообщение об ошибке вида «`at position` p`: expected` c_1[`or` c_2[`or` c_3]], `found` e». Такое сообщение означает, что первые $(p-1)$ символов строки являются ПСПП или началом какой-либо ПСПП, а на следующей позиции должен оказаться один из вариантов c_1, c_2 или c_3, чтобы строка была ПСПП или началом какой-либо ПСПП. Вместо этого на p-й позиции строки оказался вариант e. Позиции в строке нумеруются с единицы.

Во втором сообщениии вместо каждого из обозначений вариантов — c_1, c_2, c_3 и e — стоит либо один из трёх возможных символов «`(`», «`|`» и «`)`», либо строка «`END`», обозначающая конец строки. Возможные значения упорядочены следующим образом: сначала «`(`», затем «`|`», далее «`)`» и, наконец, «`END`». Значения в списке c_i должны быть перечислены в соответствующем этому порядке.

Квадратные скобки означают, что часть сообщения, заключённая в них, может присутствовать, а может отсутствовать: например, если в какой-то позиции есть ровно два правильных варианта, сообщение выглядит как «`at position` p`: expected` c_1 `or` c_2`, found` e».

Формат входных данных

Единственная строка ввода имеет длину от 1 до 60 символов и состоит исключительно из символов «`(`», «`|`» и «`)`» (ASCII-коды 40, 124 и 41).

Обратите внимание: после всех этих символов кончается сначала строка (то есть далее следует последовательность символов, обозначающая перевод строки), а потом уже файл.

Формат выходных данных

Выведите сообщение о первой неправильной позиции в строке или о том, что строка является ПСПП. Сообщение должно иметь формат, указанный в условии. Пожалуйста, соблюдайте его как можно точнее.

Примеры

errmess.in	errmess.out
`(|(|))`	`correct, length = 6`
`(|))|`	`at position 4: expected ( or END, found )`
`((`	`at position 3: expected ( or |, found END`

Пояснения к примерам

В первом примере строка «`(|(|))`» целиком является ПСПП. Она получается по пункту 3 рекурсивного определения для пустой строки A и B = `(|)`.

Во втором примере строка «`(|)`» или, например, «`(|)(|)`» была бы ПСПП. Однако нет ПСПП, начинающейся на «`(|))`». Из возможных вариантов открывающая скобка «`(`» должна быть выведена раньше конца строки «`END`».

В третьем примере строка могла бы продолжаться как «`((|)|)`» или «`(((|)|)|)`». Вместо этого на позиции 3 оказался конец строки, обозначаемый при выводе сообщения как «`END`». Из возможных вариантов открывающая скобка «`(`» должна быть выведена раньше вертикальной черты «`|`».

Задача C. Поиск

Имя входного файла:	`find.in`
Имя выходного файла:	`find.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В этой задаче нужно уметь выяснять, содержится ли число в последовательности.

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и k ($1 \leqslant n \leqslant 5000$, $1 \leqslant k \leqslant 300\,000$). Во второй строке задана последовательность из n целых чисел a_1, a_2, …, a_n, записанных через пробел ($1 \leqslant a_i \leqslant 10^9$). В третьей строке записаны запросы — k целых чисел b_1, b_2, …, b_k через пробел ($1 \leqslant b_j \leqslant 10^9$).

Формат выходных данных

В выходной файл выведите k строк. В j-ой строке выведите "`YES`", если число b_j содержится в последовательности $\{a_i\}$, и "`NO`" в противном случае.

Примеры

find.in	find.out
3 3 2 3 5 1 2 3	NO YES YES
3 4 2 1 2 2 4 1 5	YES NO YES NO
5 1 11111 1111 111 11 1 12345	NO

Задача D. Дубликатор

Имя входного файла:	`multiply.in`
Имя выходного файла:	`multiply.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Зина собрала удивительное устройство — дубликатор. Если положить в рабочий отсек какие-то вещи, в приёмник запихнуть k конфет, а после этого нажать кнопку, то из каждой вещи в рабочем отсеке получится k её точных копий. Например, если конфета была всего одна, ничего не случится, а если конфеты было две, каждая вещь раздвоится. Процедуру дубликации можно производить сколько угодно раз, главное — каждый раз запихивать в приёмник конфеты, ведь без конфет вещи в рабочем отсеке просто исчезнут! Есть и ограничение — дубликатор отказывается копировать конфеты.

Узнав об этом, Петя решил исполнить свою давнюю мечту: набрать столько кирпичиков Лего, чтобы построить из них целый дом. По его подсчётам, для постройки дома нужно не менее n кирпичиков. У Пети в кармане нашёлся один кирпичик Лего, но — вот беда! — ни одной конфеты.

Петя пошёл в буфет, чтобы купить конфеты для дубликатора. Помогите ему разработать такой план использования дубликатора, чтобы из одного кирпичика Лего в итоге получилось не менее n кирпичиков, а конфет при этом пришлось купить как можно меньше.

Формат входных данных

В первой строке входных данных записано целое число n — минимальное желаемое количество кирпичиков ($2 \leqslant n \leqslant 1\,000\,000\,000$).

Формат выходных данных

В первой строке выведите план использования дубликатора в формате «k_1 `*` k_2 `*` … `*` k_m». Здесь величины k_1, k_2, …, k_m и их количество m — целые положительные числа. Такое выражение означает, что Петя кладёт свой кирпичик в рабочий отсек, после чего запихивает в приёмник k_1 конфет и нажимает кнопку, затем запихивает в приёмник k_2 конфет и опять нажимает кнопку, и так далее. По окончании этой процедуры количество кирпичиков в рабочем отсеке должно быть не меньше n.

Если существует несколько планов, для которых потребуется купить минимально возможное количество конфет, можно вывести любой из них.

Примеры

multiply.in	multiply.out
5	5
7	3 * 3

Пояснения к примерам

В первом примере Пете нужно хотя бы пять кирпичиков для постройки дома. Один из оптимальных ответов — запихнуть в приёмный отсек пять конфет и нажать на кнопку. Как видно, для этого потребуется пять конфет. Пример другого правильного ответа — «`3 * 2`», при этом кирпичиков получается шесть. Четырьмя конфетами обойтись не получится.

Во втором примере нужно получить не меньше семи кирпичиков. Пяти конфет уже недостаточно, а вот шести хватит. Например, можно запихнуть в приёмный отсек три конфеты, нажать на кнопку, запихнуть в приёмный отсек ещё три конфеты и снова нажать на кнопку. Кирпичиков получится девять. Пример другого правильного ответа — «`2 * 2 * 2`», при этом кирпичиков будет восемь.

Задача E. Зоны патрулирования

Имя входного файла:	`patrol-zones.in`
Имя выходного файла:	`patrol-zones.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Прямой участок границы разделён на n участков километровой длины, некоторые из которых считаются непроходимыми. Нужно разбить все проходимые участки на зоны патрулирования. Каждая зона патрулирования должна состоять из двух или трёх проходимых участков. Кроме того, расстояние между любыми двумя участками одной зоны патрулирования в километрах должно быть строго меньше, чем количество участков в ней. Расстоянием между участками считается расстояние между ближайшими их точками. В частности, расстояние между соседними участками равно нулю.

Сколько существует способов разбить проходимую часть границы на зоны патрулирования?

Формат входных данных

Первая строка ввода содержит целое число n — длина границы в километрах ($1 \leqslant n \leqslant 50$). Вторая строка описывает границу и состоит из n символов из набора {«`.`» (точка), «`X`» (большая буква икс)}. Символ «`.`» соответствует проходимым участкам границы, а символ «`X`» — непроходимым.

Формат выходных данных

Выведите одно число — количество способов разбить границу на зоны патрулирования.

Примеры

patrol-zones.in	patrol-zones.out
5 `.....`	4
5 `...X.`	1
9 `...X.X...`	2
6 `.XX...`	0

Пояснения к примерам

В первом примере есть четыре способа разбить границу на две зоны патрулирования (обозначим их буквами `A` и `B`): `AAABB`, `AABAB`, `AABBB` и `ABABB`.

Во втором примере способ всего один: `AABXB`.

В третьем примере есть всего два способа разбить границу на три зоны патрулирования `A`, `B` и `C`: `AAAXBXBCC` и `AABXBXCCC`.

В четвёртом примере первый участок не может оказаться ни в какой зоне патрулирования, поэтому ни одного разбиения не существует.

Задача F. Простейшее преобразование

Имя входного файла:	`simplest.in`
Имя выходного файла:	`simplest.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Пусть $\mathcal{S}$ — строка, состоящая только из строчных букв латинского алфавита. Назовём *простым преобразованием* $\mathcal{S}$ замену каждого символа этой строки на какой-то другой символ. *Простотой* такого преобразования будем считать количество таких позиций в строке, что алфавитные номера старой и новой букв, стоящих на этой позиции, отличаются на простое число. *Простейшим* будем считать преобразование, простота которого максимальна среди всех простых преобразований исходной строки.

К примеру, если исходная строка — "`abc`", то замена '`a`' на '`c`', '`b`' на '`a`' и '`c`' на '`f`' будет простым преобразованием этой строки; поскольку номера старых и новых букв отличаются на 2, 1 и 3, соответственно, простота этого преобразования равна двум. Преобразование, однако, не является простейшим, поскольку замена '`a`' на '`c`', '`b`' на '`d`' и '`c`' на '`a`' также является простым преобразованием и имеет простоту 3.

По данной строке $\mathcal{S}$ найдите результат её простейшего преобразования. Если возможных простейших преобразований несколько, выберите то, результат которого лексикографически минимален как строка.

Формат входных данных

В первой строке входного файла задана строка $\mathcal{S}$. Она непуста, содержит только строчные буквы латинского алфавита и имеет длину не более 100 символов.

Формат выходных данных

В первой строке выходного файла выведите лексикографически наименьший результат простейшего преобразования данной строки.

Примеры

simplest.in	simplest.out
abc	cda
fade	acab

Задача G. Билеты в кино

Имя входного файла:	`tickets.in`
Имя выходного файла:	`tickets.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Петя и Миша решили пойти в кино. Но, чтобы пойти в кино, надо купить билеты, а два билета стоят n рублей. Ребята решили поделить между собой расходы на билеты, сыграв в следующую игру.

У Пети и Миши есть монеты k типов, достоинством r_1, r_2, …, r_k рублей. Они по очереди кладут по одной монете любого типа на стол; начинает Петя. Как только на столе оказывается сумма, достаточная для приобретения билетов, тот из ребят, чья очередь класть монету сейчас наступила, объявляется проигравшим и должен дополнительно купить другому мороженое. Количество монет каждого типа у обоих игроков будем считать неограниченным.

Кто выиграет при правильной игре?

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и k — требуемая сумма и количество типов монет, соответственно ($0 \leqslant n \leqslant 10\,000$, $1 \leqslant k \leqslant 10$). Во второй строке заданы k целых чисел r_1, r_2, …, r_k через пробел — достоинства монет ($1 \leqslant r_i \leqslant 100$).

Формат выходных данных

В первой строке выходного файла выведите «`PETYA`», если при правильной игре мороженое достанется Пете, и «`MISHA`», если Мише.

Примеры

tickets.in	tickets.out
10 3 1 2 5	PETYA
2 1 1	MISHA
21 7 6 5 4 3 3 2 1	MISHA

Задача H. Вечерняя прогулка

Имя входного файла:	`walk.in`
Имя выходного файла:	`walk.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

После трудного дня герцог Фред решил отвести своего друга, барона Питера, на прогулку в сад. В саду у герцога много примечательных мест — пруд, беседка, трёхсотлетний дуб, посаженный ещё пра-пра-пра-...-прадедом нынешнего герцога... Между некоторыми из этих мест есть тропинки.

Герцог хочет провести в саду не менее t минут — иначе гость не успеет нагулять аппетит перед ужином. С другой стороны, если гулять слишком долго, то наступит ночь, похолодает, и престарелый барон может простудиться, поэтому не следует гулять по саду более T минут.

Герцог Фред знает, сколько времени займёт перемещение по каждой тропинке от одного её конца до другого. Он стал придумывать маршрут, по которому следует совершить прогулку. Для этого он сперва пронумеровал все интересные места в своём саду числами от 1 до N. Маршрут должен начаться и закончиться в месте с номером 1 — входе в герцогский дом. Кроме того, маршрут должен проходить через все остальные примечательные места не более одного раза. И наконец, время прогулки, равное суммарному времени прохода по всем тропинкам маршрута, должно быть от t до T минут, включительно.

Размышляя о маршруте, герцог спросил себя, а сколько всего маршрутов, удовлетворяющих всем перечисленным ограничениям, существует. Тут герцог Фред вынужден был констатировать, что его способностей к вычислению явно не хватает, чтобы ответить на этот вопрос. Помогите герцогу посчитать, сколько всего существует таких маршрутов, и... как знать, может быть, вы из подмастерья станете его придворным математиком.

Формат входных данных

В первой строке входного файла заданы целые числа N, M, t и T через пробел ($2 \leqslant N \leqslant 10$, $0 \leqslant M \leqslant 45$, $1 \leqslant t \leqslant T \leqslant 10\,000$). Следующие M строк содержат по три числа u_i v_i l_i каждая ($1 \leqslant u_i, v_i \leqslant N$, $1 \leqslant l_i \leqslant 1000$); эти три числа означают, что в саду есть тропинка между примечательными местами с номерами u_i и v_i, и чтобы по ней пройти в любую сторону, требуется l_i минут. Между любыми двумя примечательными местами может быть не более одной тропинки. В саду нет тропинок, соединяющих какое-то примечательное место с ним самим.

Формат выходных данных

Выведите в выходной файл одно число — количество маршрутов, удовлетворяющих всем ограничениям, поставленным герцогом. Маршруты считаются различными, если последовательности примечательных мест, посещаемых на этих маршрутах, различаются.

Примеры

walk.in	walk.out
2 1 5 7 1 2 3	1
2 1 3 5 1 2 3	0
3 2 1 10 1 2 1 2 3 1	1
3 3 1 10 1 2 1 2 3 2 3 1 3	4