A. Неравенство

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

С целью автоматизации проверки усвоения материала учащимися при обучении могут создаваться специальные программы (тестирующие системы). Комплексные системы состоят из множества специальных функций, модулей, классов, каждый из которых находит правильный ответ для какого-то класса задач. Также в системе обычно имеется отдельная универсальная система сверки полученных от учащихся ответов с результатами, полученными от этих специализированных модулей-решателей.

Вам предлагается написать программу, которая будет выдавать правильные ответы на математическую задачу, которую дают школьникам.

Условие задачи: пусть имеется неравенство $a \le x \le b$, тогда требуется определить A и B в неравенстве $A \le x^2 \le B$.

Входные данные

Через пробел даны два целых числа a ($-1000 \le a \le b$) и b ($a \le b \le 1000$).

Выходные данные

Через пробел два целых числа A и B – ответ на задачу.

Примеры

Ввод	Вывод
2 3	4 9

B. Конкатенация строк

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Конкатенация (склейка) строк очень частая операция, используемая в программировании. Сейчас, наверное, уже почти все языки программирования поддерживают эту операцию, и её реализация программистом не требуется, но так было не всегда.

Пусть операция конкатенации в некотором скриптовом языке программирования реализована посредством оператора «+» и выгладит так: variable_1 = variable_2 + variable_3. Задание начальных значений переменных реализовано в виде: variable_2 = alfa_1.

Здесь в примере variable_1, variable_2 и variable_3 – имена переменных, знак «=» – оператор присваивания, а «+» – знак операции конкатенации.

Требуется написать программу, которая будет анализировать строку за строкой программы, и для каждой операции конкатенации будет выводить на отдельной строке результат конкатенации (не более 1000 первых символов результата).

Имена переменных и их значения могут состоять только из строчных и прописных латинских букв, знака подчёркивания и цифр. Имена переменных не могут начинаться на цифру.

Входные данные

В первой строке дано целое число N ($2 \le N \le 20$) – количество строчек кода. Далее в каждой из N строк одна из записей: либо присвоение переменной заданной строки, либо присвоение строке результата конкатенации значений двух переменных. Гарантируется,

что в операции конкатенации участвуют только заданные ранее переменные, длина любой строки кода не превосходит 1000 символов и имеется хотя бы одна строка кода, производящая операцию конкатенации.

Выходные данные

Для каждой конкатенации выведите на отдельной строке результат конкатенации, но не более 1000 символов, то есть если длина результат превысит 1000 символов, необходимо вывести только первую тысячу.

Примеры

Ввод	Вывод
6 hello = HeLLo world = World text = hello + world text = text + hello text = 1 hello = text + hello	HeLLoWorld HeLLoWorldHeLLo 1HeLLo

C. Простые числа

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Одними из важнейших математических исследований являются те, которые направлены на исследование свойств простых чисел. Так в данной задаче нужно написать программу, которая будет определять, есть ли среди имеющихся чисел такие, которые делят друг друга без остатка.

Входные данные

В первой строке дано целое число N ($2 \le N \le 10$) – количество простых чисел во входных данных. Далее в N строках, по одному на строку, даны простые числа p_i ($2 \le p_i \le 10^{100}$).

Выходные данные

Выведите «YES» (без кавычек), если есть хотя бы одна пара чисел во входных данных, что одно число делит другое без остатка и «NO» (без кавычек) в противном случае.

Примеры

Ввод	Вывод
3 13 2 7	NO

D. Циклический сдвиг и модификация

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Имеется битовая последовательность, состоящая из L бит. Циклический сдвиг битовой последовательности влево на одну позицию приводит к тому, что самый левый бит

убирается слева и помещается справа. Циклический сдвиг на X бит, равносилен повторению X раз операции сдвига на одну позицию.

Модификация одного из бита в позиции P указывает, что бит, стоящий по порядку слева (счёт с единицы начинается) на позиции P меняет своё состояние на противоположное.

Требуется после каждого действия: либо циклический сдвиг на X бит, либо модификация одного бита, давать ответ на вопрос: была ли в начале или получена после завершения каждой из операций на предыдущих шагах в точности такая же битовая последовательность.

Входные данные

В первой строке дана битовая последовательность длины L ($1 \le L \le 10^4$). На следующей строке дано одно целое число N ($1 \le N \le 10^4$) – количество действий над последовательностью. Далее идёт N строк, в каждой описание одного действия. Первый знак «X» (без кавычек) говорит, что произведён циклический сдвиг, и затем через пробел от знака указана величина сдвига x ($0 \le x \le 10^9$). Знак «P» (без кавычек) указывает, что значение через пробел в этой же строке указывает номер модифицированного бита p ($1 \le p \le L$).

Выходные данные

Для каждого действия необходимо вывести одну строку с ответом на вопрос: «была ли изначально или получена при модификациях в точности такая же строка?». Соответственно выведите «YES» (без кавычек) или «NO» (без кавычек).

Примеры

Ввод	Вывод
1001 4 P 2 X 1 P 3 P 2	NO NO YES YES

Пояснение к примеру

Изначально имеем битовую последовательность 1001.

После операции P 2 (модификация бита на 2-й позиции) имеем строку 1101, таких строк ещё не было, поэтому выводим «NO».

Затем, после операции X 1 (сдвиг влево на 1 позицию), получаем строку 1011, таких строк ещё не было, поэтому выводим «NO».

И после очередной операции, в данном случае P 3 (модификация бита на 3-й позиции) имеем строку 1001, такая строка была в самом начале, поэтому выводим «YES».

И завершает последняя операция P 2 (модификация бита на 2-й позиции) – имеем строку 1101, то есть в точности как после первой операции, поэтому выводим «YES».

E. Битовые операции

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Битовые операции циклического сдвига, операции логического побитового OR, AND, XOR поддерживаются на уровне процессора и востребованы во многих задачах, решаемых на компьютере.

Вам будет дано одно 16 битное целое положительное число, записанное в десятичной системе счисления. И даны операции над этим числом в двоичном представлении: ROR (циклический сдвиг вправо на один двоичный знак), ROL (циклический сдвиг влево на один двоичный знак), XOR N (побитовая операция XOR с числом N, которое задано в десятичной системе счисления), OR N (побитовая операция OR с числом N, которое задано в десятичной системе счисления) и AND N (побитовая операция AND с числом N, которое задано в десятичной системе счисления). Требуется вывести двоичное представление результата после каждой операции-модификатора.

Входные данные

В первой строке дано целое число A ($0 \le A < 2^{16}$) – начальное значение. На следующей строке дано одно целое число N ($1 \le N \le 10$) – количество действий над последовательностью. Далее в N строках, по одной операции на строку. На величины значений N при операциях XOR, OR и AND имеются ограничения аналогичные ограничению начального значения A. Обратите внимание, что все числа 16 битные!

Выходные данные

После каждой операции выведите на отдельной строке текущее значение величины в двоичном представлении, то есть по 16 двоичных цифр (0, 1) в каждой строке.

Примеры

Ввод	Вывод
65533 5 ROR ROL OR 2 AND 65533 AND 65534	1111111111111110 1111111111111101 1111111111111111 1111111111111101 1111111111111100

F. Ошибка в коде

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Когда учащиеся школ сдают ЕГЭ по информатике, то в части заданий может встретиться потребность модифицировать предложенный код (исправить ошибку). Затем эти работы вручную сверяются. Кроме того на каком-то этапе эти задачи готовятся. На всех этапах может быть полезно иметь автоматическую систему проверки кода.

Так как программа может быть написана на различных языках, то представляется эффективным написание программы сверки не анализируя код, а по выданным программой данным на множестве тестов.

Здесь требуется написать программу, которая по выданным программой данным автоматически будет находить и указывать сделанную в коде ошибку, исходя из предположения, что ошибка могла быть только в одном из ожидаемых мест.

Далее дан алгоритм, описанный на си, в «/* */» (без кавычек) описаны возможные соответствующие ошибки. В самом изложенном коде дан правильный алгоритм. После «//» даны однострочные комментарии и ограничение на входные данные.

Алгоритм, описанный на си, и места возможных ошибок

```
int n;
int s = 0; /* здесь может быть int s = 1 */
scanf("%d", &n); // Ввод целого числа в переменную n (0 ≤ n ≤ 1000)
for(int i = 1; i <= n; ++i) /* i < n или i > n или --i */
    s = s + i; /* здесь может быть s = s - i */
printf("%d", s);
```

Тип данных int – 32 битное целое число со знаком.

Входные данные

В первой строке дано целое число N ($1 \le N \le 5$) – количество произведённых тестов. Далее в N строках по одному числу – ответ выданный программой на какие-то входные данные.

Выходные данные

Выведите код программы без ошибок или с одной из возможных ошибок, который мог бы дать такие ответы, как представленные на входе, учитывая, что входные данные соответствуют ограничениям, данным в комментарии к коду (см. значение, вводимое в переменную n).

Код представьте без комментариев, то есть без содержания «/* */» и «//». Отступ строки в цикле выполните в виде 4-х пробелов. Пробелы расставьте как в задании. Все строчки не должны содержать дополнительных пробелов в начале и в конце строки.

Если есть несколько правильных ответов и один из них код без ошибок, выведите код без ошибок. Если есть несколько вариантов ответов, и среди них нет варианта кода без ошибки, то выведите слово «ERROR».

Примеры

Ввод	Вывод

1
10
	int n;
	int s = 0;
	scanf("%d", &n);
	for(int i = 1; i <= n; ++i)
	    s = s + i;
	printf("%d", s);

1
-10
	int n;
	int s = 0;
	scanf("%d", &n);
	for(int i = 1; i <= n; ++i)
	    s = s - i;
	printf("%d", s);

G. Условие Фано

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Пусть имеется набор различных символов (знаков), являющийся подмножеством символов английского алфавита (ABCDEFGHIJKLMNOPQRSTUVWXYZ). Каждому знаку этого множества может быть сопоставлена последовательность, состоящая только из нулей и единиц.

Если при сопоставлении битовых последовательностей потребовать, чтобы ни одна такая последовательность не являлась началом (префиксом) другой (соблюдалось условие Фано), то текст, изложенный в виде этих последовательностей, гарантированно будет самотерминирующимся (будет известно, где заканчивается один символ и начинается другой).

Входные данные

В первой строке дано число N ($2 \le N \le 26$) – количество различных символов в наборе. Далее имеется N строк, каждая из которых вначале содержит символ набора и через пробел, если кодовая последовательность определена для этого символа, последовательность из m_i ($1 \le m_i \le 10$) нулей и единиц (собственно, кодовая последовательность, сопоставленная этому знаку английского алфавита). Гарантируется, что кодовая последовательность не определена хотя бы для одного знака и данные кодовые последовательности составлены с учётом условия Фано.

Выходные данные

Выведите одну двоичную последовательность минимальной длины – допустимый код для любого из знаков, для которого нет кода во входных данных. Обратите внимание, что с учётом данного вами кода должна быть возможность определить кодовые последовательности для всех знаков набора и при этом соблюсти условие Фано. Если есть несколько решений, то выведите то, которое лексикографически меньше. Если решения не существует, то выведите «impossible» (без кавычек).

Примеры

Ввод	Вывод
3 A 10 B 11 C	0
4 Z A 10 B 11 C	00
4 Z 0 A 10 B 11 C	impossible

H. Геометрия циферблата

Ограничение по времени: **1000 миллисекунд**
Ограничение по памяти: **65000 кибибайт**

Один из возможных механизмов часов может работать по следующему алгоритму:

- когда перемещается секундная стрелка, другие стрелки стоят на месте, кроме, когда секундная стрелка перемещается с позиции «59» секунд в позицию «0» секунд, в это время минутная, абсолютно с той же скоростью в каждый момент времени перемещается на одно деление (1/60 часть циферблата);

- секундная стрелка перемещается от одного деления (1/60 часть циферблата) к другому ровно за одну секунду;

- часовая стрелка перемещается на одно деление (1/60 часть циферблата) только в тот момент, когда минутная отсчитывает очередные 12 минут (1/5 циферблата), при этом она также перемещается одновременно и с той же скоростью что и минутная в течение этой секунды. То есть, например, когда минутная стрелка перемещается с позиции 11 минут на позицию 12 минут, в эту секунду и часовая стрелка перемещается на одно деление.

Напишите программу, которая для заданного интервала времени определит, сколько времени в секундах минутная и часовая стрелки (которые принять для решения математическими отрезками, исходящими из центра циферблата) находятся друг к другу строго под прямым углом (90 градусов).

Гарантируется, что заданный интервал времени меньше длины суток и не менее 1 секунды.

Входные данные

В первой строке время, определяющее начало интервала времени, во второй строке его конец. Формат, задающий момент времени: чч:мм:сс, где чч (0..23), мм (00..59), сс (00..59).

Выходные данные

Одно целое число – количество секунд, в течение которых минутная и часовая стрелки на указанном интервале находились строго под углом 90 градусов.

Примеры

Ввод	Вывод
03:15:23 14:13:01	1240
18:01:59 03:12:00	1123
03:00:00 03:01:00	59
08:59:00 09:00:00	60
03:00:59 03:32:00	0