

## Задача A. Set

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

Реализуйте множество с использованием хеш таблицы.

### Формат входных данных

Входной файл содержит описание операций. В каждой строке находится одна из следующих операций:

- **insert**  $x$  — добавить элемент  $x$  в множество. Если элемент уже есть в множестве, то ничего делать не надо.
- **delete**  $x$  — удалить элемент  $x$ . Если элемента  $x$  нет, то ничего делать не надо.
- **exists**  $x$  — если ключ  $x$  есть в множестве выведите «true», если нет «false».

В множество помещаются и извлекаются только целые числа, не превышающие по модулю  $10^9$ .

### Формат выходных данных

Выведите последовательно результат выполнения всех операций **exists**. Следуйте формату выходного файла из примера.

### Пример

set.in	set.out
insert 2	true
insert 5	false
insert 3	false
exists 2	
exists 4	
insert 2	
delete 2	
exists 2	

## Задача В. Мар

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

Реализуйте ассоциативный массив с использованием хеш таблицы.

### Формат входных данных

Входной файл содержит описание операций. В каждой строке находится одна из следующих операций:

- `put x y` — поставить в соответствие ключу `x` значение `y`. Если ключ уже есть, то значение необходимо изменить.
- `delete x` — удалить ключ `x`. Если элемента `x` нет, то ничего делать не надо.
- `get x` — если ключ `x` есть в ассоциативном массиве, то выведите соответствующее ему значение, иначе выведите «none».

Ключи и значения — строки из латинских букв длиной не более 20 символов.

### Формат выходных данных

Выведите последовательно результат выполнения всех операций `get`. Следуйте формату выходного файла из примера.

### Пример

<code>map.in</code>	<code>map.out</code>
<code>put hello privet</code>	<code>privet</code>
<code>put bye poka</code>	<code>poka</code>
<code>get hello</code>	<code>none</code>
<code>get bye</code>	
<code>delete hello</code>	
<code>get hello</code>	

## Задача С. LinkedMap

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

Реализуйте прошитый ассоциативный массив с использованием хеш таблицы.

### Формат входных данных

Входной файл содержит описание операций. В каждой строке находится одна из следующих операций:

- **put**  $x$   $y$  — поставить в соответствие ключу  $x$  значение  $y$ . Если элемент уже есть, то значение необходимо изменить.
- **delete**  $x$  — удалить ключ  $x$ . Если элемента  $x$  нет, то ничего делать не надо.
- **get**  $x$  — если ключ  $x$  есть в множестве выведите соответствующее ему значение, если нет выведите «none».
- **prev**  $x$  — вывести значение соответствующее ключу находящемуся в ассоциативном массиве, который был вставлен позже всех, но до  $x$  или «none», если такого нет или в массиве нет  $x$ .
- **next**  $x$  — вывести значение соответствующее ключу находящемуся в ассоциативном массиве, который был вставлен раньше всех, но после  $x$  или «none», если такого нет или в массиве нет  $x$ .

Ключи и значения — строки из латинских букв длиной не более 20 символов.

### Формат выходных данных

Выведите последовательно результат выполнения всех операций **get**, **prev**, **next**. Следуйте формату выходного файла из примера.

### Пример

linkedmap.in	linkedmap.out
put zero a	c
put one b	b
put two c	d
put three d	c
put four e	a
get two	e
prev two	none
next two	
delete one	
delete three	
get two	
prev two	
next two	
next four	

## Задача D. MultiMap

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

Реализуйте множественное отображение с использованием хеш таблиц.

### Формат входных данных

Входной файл содержит описание операций. В каждой строке находится одна из следующих операций:

- `put  $x$   $y$`  — добавить пару  $(x, y)$ . Если пара уже есть, то второй раз её добавлять не надо.
- `delete  $x$   $y$`  — удалить пару  $(x, y)$ . Если пары нет, то ничего делать не надо.
- `deleteall  $x$`  — удалить все пары с первым элементом  $x$ .
- `get  $x$`  — вывести количество пар с первым элементом  $x$ , а затем вторые элементы всех этих пар в произвольном порядке.

Ключи и значения — строки из латинских букв длиной не более 20 символов.

### Формат выходных данных

Выведите последовательно результат выполнения всех операций `get`. Следуйте формату выходного файла из примера. Гарантируется, что размер выходного файла не превысит 10 мегабайт.

### Пример

multimap.in	multimap.out
put a a	3 b c a
put a b	2 c a
put a c	0
get a	
delete a b	
get a	
deleteall a	
get a	

## Задача Е. Идеальное хеширование

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

Вам задана архитектура компьютера, в рамках которой от вас требуется сделать программу, которая является идеальным хешом для заданного вам множества целых чисел.

Опишем архитектуру компьютера. У него есть  $4 \text{ мегабайта} = 4 \times 2^{20}$  памяти, доступной только для чтения, содержащей программу, которую будет исполнять компьютер, и данные. Процессор компьютера содержит 256 регистров с названиями от `r0` до `r255`, каждый из них может сохранить 32-битное целое число. Перед исполнением программы значение каждого регистра кроме `r0` равно 0, а в `r0` содержится входные данные программы.

У компьютера есть специальный регистр, который называется *instruction pointer*. Программа хранится в памяти и изначально значение регистра `instruction pointer` равно 0. Выполнение программы проходит по следующему сценарию. Сначала читается номер инструкции, который хранится в памяти по адресу значения `instruction pointer`, после чего читаются аргументы инструкции и инструкция выполняется. Если во время выполнения инструкции не случился `jump`, то значение `instruction pointer` увеличивается на размер прочитанной инструкции, включая аргументы.

Программа должна быть отображением из множества заданных  $n$  целых чисел  $A = \{a_1, a_2, \dots, a_n\}$  в множество целых чисел от 0 до  $2n - 1$ . Она должна завершаться после не более чем 30 выполненных инструкций. Отображение не должно иметь коллизий. Формально, надо сделать программу в этой архитектуре, которая удовлетворяет следующим критериям:

- Если программе на вход подать одно из чисел из множества  $A$ , то она должна завершиться за не более чем 30 инструкций и вывести целое число от 0 до  $2n - 1$ .
- Для любых двух различных чисел  $a_i$  и  $a_j$  из множества  $A$  значения на выходе должны быть различны.

Вам задана некоторая документация.

- Все регистры содержат 32-битные целые числа
- Для сложения, вычитания и битовых операций не важно, числа знаковые или нет
- Команды `jump` с условием представляют числа как знаковые
- Результатом умножения или деления является знакового 64-битное число, которое сохраняется в два регистра. Регистр, который получает младшую часть результата сохраняет его как беззнаковое число
- Деление и взятие остатка принимают делимое в качестве 64-битных чисел, младшая часть интерпретируется как беззнаковое число, а старшая — как знаковое. Делитель — знаковое число.
- Деление и взятие остатка знаковых чисел производится так же, как это делается в архитектуре x86.
- Все числа в памяти сохраняются в порядке от младших байт к старшим (little endian).

От вас требуется понять, что должна содержать память, чтобы исполнялась программа, удовлетворяющая условиям, описанным выше.

Для любых входных данных из множества  $A$  ваша программа не должна пытаться читать данные извне 4МБ-диапазона и не должна делить на ноль. Значение регистра `instruction pointer` так же должно быть внутри 4МБ-диапазона.

## Таблица инструкций

Инстр.	Размер	Opcode	Что делает инструкция
nop	1 байт	00	Ничего
add	4 байта	01 r1 r2 r3	$r3 := r1 + r2$
sub	4 байта	02 r1 r2 r3	$r3 := r1 - r2$
mul	5 байт	03 r1 r2 r3 r4	$(r3, r4) := r1 * r2$ r3 получает младшие биты, а r4 — старшие биты произведения
div	6 байт	04 r1 r2 r3 r4 r5	$(r3, r4) := (r1, r5) \text{ div } r2$ r1 содержит младшие биты, r5 содержит старшие биты делимого, r3 получает младшие биты, а r4 — старшие биты частного
mod	5 байт	05 r1 r2 r3 r4	$r3 := (r1, r4) \text{ mod } r2$ , r1 содержит младшие биты, r4 содержит старшие биты делимого
and	4 байта	10 r1 r2 r3	$r3 := r1 \text{ and } r2$
or	4 байта	11 r1 r2 r3	$r3 := r1 \text{ or } r2$
xor	4 байта	12 r1 r2 r3	$r3 := r1 \text{ xor } r2$
neg	2 байта	20 r1	$r1 := -r1$
not	2 байта	21 r1	$r1 := \sim r1$
load	3 байта	30 r1 r2	$r1 := \text{memory}[r2]$ 4 байта, начиная с адреса r2 копируются в регистр r1, первым копируется младший байт
put	6 байт	31 r1 b0 b1 b2 b3	$r1 := (b0, b1, b2, b3)$ здесь b0 — младший байт числа, положенного в регистр, а b3 — старший
jmp	5 байт	40 b0 b1 b2 b3	Сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
jz	6 байт	41 r1 b0 b1 b2 b3	Если $r1 == 0$ сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
jnz	6 байт	42 r1 b0 b1 b2 b3	Если $r1 \neq 0$ сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
jg	6 байт	43 r1 b0 b1 b2 b3	Если $r1 > 0$ сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
jge	6 байт	44 r1 b0 b1 b2 b3	Если $r1 \geq 0$ сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
jl	6 байт	45 r1 b0 b1 b2 b3	Если $r1 < 0$ сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
jle	6 байт	46 r1 b0 b1 b2 b3	Если $r1 \leq 0$ сделать <b>jump</b> к инструкции (b0, b1, b2, b3)
ret	1 байт	ff	Завершить программу. Выходные данные содержатся в регистре r0

## Формат входных данных

Первая строка содержит целое число  $n$  ( $1 \leq n \leq 100\,000$ ).

Вторая строка содержит  $n$  различных целых чисел  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ).

## Формат выходных данных

Выведите данные, содержащиеся в памяти компьютера. Вам разрешается вывести любое число байт от 1 до 4194304. Остальная часть памяти заполнится нулями. Каждый байт должен быть выведен как шестнадцатеричное число из двух цифр. Не печатайте пробелы.

Пожалуйста, выводите только требующийся префикс памяти вашей программы: не выводите конечные нули, чтобы тестирование проходило быстрее.

## Примеры

стандартный ввод	стандартный вывод
3 2 3 9	3101040000000500010003ff
2 6 10	300000ff000000000000001000000

## Задача F. Хеш-код

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

Согласно документации стандартной библиотеки Java, хеш-код для строки вычисляется как

$$s[0] * 31^{(n-1)} + s[1] * 31^{(n-2)} + \dots + s[n-1]$$

Где  $s[i]$  — это  $i$ -й символ строки,  $n$  длина строки,  $\wedge$  обозначает возведение в степень. Для вычисления используются целые 32-битные числа в форме дополнения до двух.

Вы собираетесь взломать сервера одной известной компании. Чтобы вы смогли выполнить атаку, вам нужны  $k$  различных строк, которые имеют одинаковые хеш-коды. К сожалению, сервера этой компании не принимают строки запроса, содержащие буквы отличные от английских букв нижнего и верхнего регистров.

Напишите программу, которая генерирует такие строки.

### Формат входных данных

Первая строка содержит целое число  $k$  — количество необходимых строк запроса для генерации ( $2 \leq k \leq 1000$ ).

### Формат выходных данных

Необходимо вывести  $k$  различных непустых строк, каждая из которых имеет длину не более 1000 символов. Все строки должны состоять только из английских букв верхнего или нижнего регистров и иметь равный хеш-код.

## Задача G. Бункеры

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

Петя и Вася с упоением играют в шпионов. Сегодня они планируют, где будут расположены их секретные бункеры и штаб-квартира.

Пока Петя и Вася решили, что им понадобится ровно  $n$  бункеров, которые для секретности будут пронумерованы числами от 1 до  $n$ . Некоторые из них будут соединены двусторонними тоннелями, причем для надежности и секретности по тоннелям можно будет попасть из любого бункера в любой единственным образом. Петя и Вася даже решили, какие из бункеров будут соединены тоннелями, но выбрать, какой из них будет штаб-квартирой, они не могут. Мальчики хотят выбрать ее и разделить оставшиеся бункеры между собой таким образом, чтобы им досталось поровну бункеров и к штаб-квартире вело бы ровно два тоннеля: один от бункера, принадлежащего Васе, другой — от бункера, принадлежащего Пете.

Уставший Петя пошел к себе домой, а утром Вася показал ему план, на котором бункеры были обозначены точками, а тоннели отрезками. Кроме того, Вася выбрал штаб-квартиру таким образом, что нарисованный им план был симметричен относительно некоторой прямой, проходящей через точку, которая соответствовала штаб-квартире. При этом бункеры, принадлежащие Пете находились с одной стороны от этой прямой, а бункеры, принадлежащие Васе, с другой стороны.

Хотя Петя почти сразу показал Васе, что тот ошибся и не нарисовал половину бункеров, ему стало интересно, можно ли выбрать штаб-квартиру и нарисовать такой симметричный план.

### Формат входных данных

В первой строке входного файла находится одно целое число  $n$  ( $1 \leq n \leq 10^5$ ) — количество бункеров. В следующих  $n - 1$  строках находится по два целых числа  $u_i$  и  $v_i$  ( $1 \leq u_i, v_i \leq n, u_i \neq v_i$ ) — номера бункеров, которые соединяет  $i$ -й тоннель. Гарантируется, что между любыми двумя бункерами существует единственный путь.

### Формат выходных данных

В выходной файл выведите «YES», если можно выбрать штаб-квартиру и нарисовать такой план, или «NO» если это невозможно.

### Примеры

стандартный ввод	стандартный вывод
2 1 2	NO
3 1 2 2 3	YES