

## Задача А. Переворот

Имя входного файла: `reverse.in`  
Имя выходного файла: `reverse.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 64 мегабайта

Дано натуральное число  $N$  и последовательность из  $N$  элементов. Требуется вывести эту последовательность в обратном порядке.

### Формат входных данных

В первой строке входного файла записано натуральное число  $N$  ( $N \leq 10^3$ ). В следующих  $N$  строках идут  $N$  целых чисел, по модулю не превосходящих 1000, — элементы последовательности.

### Формат выходных данных

В выходной файл выведите заданную последовательность в обратном порядке.

### Примеры

| <code>reverse.in</code> | <code>reverse.out</code> |
|-------------------------|--------------------------|
| 2                       | 4                        |
| 3                       | 3                        |
| 4                       |                          |

### Замечание

Для вывода последовательности в обратном порядке необходимо использовать рекурсивную функцию.

## Задача В. Скобки

Имя входного файла: `brackets.in`  
Имя выходного файла: `brackets.out`  
Ограничение по времени: 0.5 second  
Ограничение по памяти: 64 megabytes

Требуется определить, является ли правильной данная последовательность круглых, квадратных и фигурных скобок.

### Формат входных данных

В единственной строке входного файла записано подряд  $N$  скобок ( $1 \leq N \leq 10^5$ ).

### Формат выходных данных

В выходной файл вывести «YES», если данная последовательность является правильной, и «NO» в противном случае.

### Примеры

| <code>brackets.in</code> | <code>brackets.out</code> |
|--------------------------|---------------------------|
| <code>()</code>          | YES                       |
| <code>([])</code>        | YES                       |

### Замечание

Скобочная последовательность называется правильной, если ее можно получить из какого-либо математического выражения вычеркиванием всех символов, кроме скобок.

Формальное определение правильной скобочной последовательности таково: 1. Пустая последовательность является правильной. 2. Если  $A$  – правильная скобочная последовательность, то  $(A)$ ,  $[A]$  и  $\{A\}$  – правильные скобочные последовательности. 3. Если  $A$  и  $B$  – правильные скобочные последовательности, то  $AB$  – правильная скобочная последовательность.

## Задача С. Постфиксная запись

Имя входного файла: postfix.in  
Имя выходного файла: postfix.out  
Ограничение по времени: 1 second  
Ограничение по памяти: 64 megabytes

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел  $A$  и  $B$  записывается как  $A B +$ . Запись  $B C + D *$  обозначает привычное нам  $(B+C)*D$ , а запись  $A B C + D * +$  означает  $A+(B+C)*D$ . Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

### Формат входных данных

В единственной строке записано выражение в постфиксной записи, содержащее однозначные числа и операции  $+$ ,  $-$ ,  $*$ . Строка содержит не более 100 чисел и операций.

### Формат выходных данных

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений по модулю меньше  $2^{31}$ .

### Примеры

| postfix.in    | postfix.out |
|---------------|-------------|
| 8 9 + 1 7 - * | -102        |

## Задача D. Грузовики

Имя входного файла: `trucks.in`  
Имя выходного файла: `trucks.out`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 64 мегабайта

Необходимо поместить несколько ящиков в грузовики. Для этого планируется разделить имеющиеся ящики на две одинаковые группы (в случае нечетного количества получаются две группы, в одной из которых на один ящик больше), потом разделить каждую из этих групп аналогичным образом, и так далее, пока мы не получим группы, которые влезают в грузовик. Как только какая-то из получившихся групп влезает в грузовик, производится загрузка ящиков этой группы, и грузовик уезжает. Требуется определить, сколько грузовиков потребуется, чтобы увезти все ящики.

### Формат входных данных

Во входном файле два числа  $n$  и  $k$  ( $2 \leq n \leq 10\,000, 1 \leq k \leq n-1$ ) — количество ящиков и емкость грузовика.

### Формат выходных данных

Выведите требуемое количество грузовиков.

### Примеры

| <code>trucks.in</code> | <code>trucks.out</code> |
|------------------------|-------------------------|
| 14 3                   | 6                       |
| 15 1                   | 15                      |
| 1024 5                 | 256                     |

## Задача Е. Парикмахерская

Имя входного файла: `saloon.in`  
Имя выходного файла: `saloon.out`  
Ограничение по времени: 1 second  
Ограничение по памяти: 64 megabytes

В парикмахерской работает один мастер. Он тратит на одного клиента ровно 20 минут, а затем сразу переходит к следующему, если в очереди кто-то есть, либо ожидает, когда придет следующий клиент.

Даны времена прихода клиентов в парикмахерскую (в том порядке, в котором они приходили).

Также у каждого клиента есть характеристика, называемая *степенью нетерпения*. Она показывает, сколько человек может максимально находиться в очереди перед клиентом, чтобы он дождался своей очереди и не ушел раньше. Если в момент прихода клиента в очереди находится больше людей, чем степень его нетерпения, то он решает не ждать своей очереди и уходит. Клиент, который обслуживается в данный момент, также считается находящимся в очереди.

Требуется для каждого клиента указать время его выхода из парикмахерской.

### Формат входных данных

В первой строке вводится натуральное число  $N$ , не превышающее 100 — количество клиентов.

В следующих  $N$  строках вводятся времена прихода клиентов — по два числа, обозначающие часы и минуты (часы — от 0 до 23, минуты — от 0 до 59) и степень его нетерпения (неотрицательное целое число не большее 100) — максимальное количество человек, которое он готов ждать впереди себя в очереди. Времена указаны в порядке возрастания (все времена различны).

Гарантируется, что всех клиентов успеют обслужить до полуночи.

Если для каких-то клиентов время окончания обслуживания одного клиента и время прихода другого совпадают, то можно считать, что в начале заканчивается обслуживание первого клиента, а потом приходит второй клиент.

### Формат выходных данных

В выходной файл выведите  $N$  пар чисел: времена выхода из парикмахерской 1-го, 2-го, ...,  $N$ -го клиента (часы и минуты). Если на момент прихода клиента человек в очереди больше, чем степень его нетерпения, то можно считать, что время его ухода равно времени прихода.

### Примеры

| saloon.in | saloon.out |
|-----------|------------|
| 3         | 10 20      |
| 10 0 0    | 10 40      |
| 10 1 1    | 10 2       |
| 10 2 1    |            |
| 5         | 1 20       |
| 1 0 100   | 2 20       |
| 2 0 0     | 2 1        |
| 2 1 0     | 2 40       |
| 2 2 3     | 2 3        |
| 2 3 0     |            |

## Задача F. Очередь

Имя входного файла: `queue.in`  
Имя выходного файла: `queue.out`  
Ограничение по времени: 3 секунды  
Ограничение по памяти: 256 мегабайт

Гоблины Мглистых гор очень любят ходить к своим шаманам. Так как гоблинов много, к шаманам часто образуются очень длинные очереди. А поскольку много гоблинов в одном месте быстро образуют шумную толпу, которая мешает шаманам производить сложные медицинские манипуляции, последние решили установить некоторые правила касательно порядка в очереди.

Обычные гоблины при посещении шаманов должны вставать в конец очереди. Привилегированные же гоблины, знающие особый пароль, встают ровно в ее середину, причем при нечетной длине очереди они встают сразу за центром.

Так как гоблины также широко известны своим непочтительным отношением ко всяческим правилам и законам, шаманы попросили вас написать программу, которая бы отслеживала порядок гоблинов в очереди.

### Формат входных данных

В первой строке записано одно целое число  $N$  ( $1 \leq N \leq 10^5$ ) — число запросов к вашей программе. В следующих  $N$  строках заданы описания запросов в следующем формате:

- «+  $i$ » — к очереди присоединяется гoblin  $i$  ( $1 \leq i \leq N$ ) и встает в ее конец;
- «\*  $i$ » — привилегированный гoblin  $i$  встает в середину очереди ( $1 \leq i \leq N$ );
- «-» — гoblin выходит из очереди и заходит к шаманам. Гарантируется, что на момент каждого такого запроса очередь будет не пуста.

### Формат выходных данных

Для каждого запроса третьего типа в отдельной строке выведите номер гоблина, который должен зайти к шаманам.

### Примеры

| queue.in | queue.out |
|----------|-----------|
| 7        | 1         |
| + 1      | 2         |
| + 2      | 3         |
| -        |           |
| + 3      |           |
| + 4      |           |
| -        |           |
| -        |           |
| 10       | 1         |
| + 1      | 3         |
| + 2      | 2         |
| * 3      | 5         |
| -        | 4         |
| + 4      |           |
| * 5      |           |
| -        |           |
| -        |           |
| -        |           |
| -        |           |