

## Задача 1. Перевод из инфиксной формы записи в постфиксную

Источник:	основная
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Необходимо написать программу, переводящую выражение, записанное в инфиксной форме, в постфиксную (обратную польскую) запись. Операндами в выражении могут быть только маленькие латинские буквы. Операции —  $+$ ,  $^$ ,  $*$ ,  $/$ .

### Формат входных данных

Входной файл содержит одну строку, в которой записано выражение в инфиксной форме. Ее длина не превышает 1000 символов.

### Формат выходных данных

В выходной файл нужно выдать это выражение в обратной польской записи.

### Пример

input.txt	output.txt
a*b-c/(d+e)	ab*cde+/-

## Задача 2. Вычисление на стеке

Источник: основная  
Имя входного файла: `input.txt`  
Имя выходного файла: `output.txt`  
Ограничение по времени: 1 секунда  
Ограничение по памяти: разумное

По заданному выражению в обратной польской записи посчитать его значение.

### Формат входных данных

В единственной строке входного файла записано выражение в обратной польской записи. Операнды — целые числа. Числа и знаки операций разделены пробелами. Длина строки не превосходит 2000.

### Формат выходных данных

В выходной файл нужно вывести одно целое число — значение выражения. Гарантируется, что деление на 0 отсутствует.

### Пример

<code>input.txt</code>	<code>output.txt</code>
12 5 + 7 - 2 10 * -	-10

## Задача 3. Циклический буфер

Источник:	основная
Имя входного файла:	<code>input.txt</code>
Имя выходного файла:	<code>output.txt</code>
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Реализация кольцевого буфера (очереди на массиве). Необходимо написать программу, моделирующую систему асинхронного сбора данных. Экспериментальные данные берутся из файла, содержащего целые числа.

В том случае, если считано положительное число  $R$ , то следующие  $R$  чисел относятся к экспериментальным данным, которые необходимо записать в кольцевой буфер (в конец очереди).

В том случае, если встретилось отрицательное число  $K$ , то необходимо забрать  $-K$  первых ячеек данных из буфера и распечатать их среднее арифметическое. В случае, когда буфер переполнился, или в нем не оказалось данных, необходимо выдать диагностическое сообщение и продолжить работу.

### Формат входных данных

В первой строке входного файла записаны через пробел два положительных целых числа  $N$  и  $M$  — длина буфера и количество элементов в управляющей программе ( $1 \leq N \leq 1000, 1 \leq M \leq 10000$ ).

В следующих строках записано по одному целому числу — управляющие команды и экспериментальные данные. Все числа не превосходят по модулю 10000.

### Формат выходных данных

В выходной файл нужно вывести результат выполнения управляющей программы.

Если в управляющей программе встретилось положительное число  $R$ , то в конец буфера записать  $R$  чисел, которые идут во входном файле после считанного числа  $R$ . Если в буфере не хватает места для всех  $R$  чисел, то записать столько, сколько влезет, и выдать в выходной файл сообщение **Memory overflow**, а те числа, которые не влезли в буфер, никуда не записывать.

Если считано в управляющей программе отрицательное число  $K$ , то удалить из буфера первые  $-K$  чисел, посчитать их среднее арифметическое (целое) и выдать его в выходной файл. Если в буфере меньше, чем  $-K$  элементов, то выдать среднее арифметическое тех, что есть, их удалить, а затем выдать сообщение **Empty queue**. Все сообщения выдавать по одному на отдельной строке.

## Пример

input.txt	output.txt
5 16	1
2	Memory overflow
1	0
-3	3
3	Empty queue
5	
1	
-4	
-4	
5	
1	
2	
3	
4	
5	
-3	
-3	

## Задача 4. Тасовка перфокарт

Источник:	основная*
Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	5 секунд
Ограничение по памяти:	специальное

Имеется две колоды перфокарт: левая колода и правая. Изначально в каждой колоде ровно  $N$  перфокарт. В левой колоде перфокарты пронумерованы числами от 1 до  $N$  по порядку, если просматривать их сверху вниз. В правой колоде перфокарты пронумерованы числами от  $-1$  до  $-N$  по порядку, если просматривать их сверху вниз.

Для перемешивания колод нужно выполнить  $M$  заданных операций, каждая операция заключается в перекалывании одной карты. Каждая операция обозначается одной шестнадцатеричной цифрой в диапазоне от 0 до F (15) включительно. Операция определяется значениями четырёх битов в двоичной записи этой цифры:

- Если старший бит (8) единичный, то нужно взять карту из правой колоды, а иначе — из левой колоды.
- Если предпоследний бит (4) единичный, то нужно взять карту снизу колоды, а иначе — сверху колоды.
- Если второй бит (2) единичный, то нужно положить карту в правую колоды, а иначе — в левую.
- Если младший бит (1) единичный, то нужно положить карту в колоду снизу, а иначе — сверху.

Если в какой-то момент нужно выполнить операцию, которая предписывает взять карту из пустой колоды, то такую операцию нужно пропустить, ничего не делая.

### Формат входных данных

В первой строке входного файла записано два целых числа:  $N$  — начальное количество карт в каждой колоде и  $M$  — сколько операций нужно выполнить ( $1 \leq N \leq 5 \cdot 10^5$ ,  $0 \leq M \leq 5 \cdot 10^6$ ).

Во второй строке записано подряд ровно  $M$  символов — описание операций в порядке их выполнения. Каждый символ является шестнадцатеричной цифрой и изменяется в диапазоне от 0 до 9 или от A до F включительно.

### Формат выходных данных

После выполнения всех операций требуется вывести содержимое левой колоды в первую строку выходного файла, и содержимое правой колоды — во вторую строку.

В каждой строке сначала должно быть записано целое число  $K$  — количество карт в колоде после выполнения всех операций, а затем через пробел  $K$  целых чисел — номера перфокарт в колоде, перечисленные в порядке сверху вниз.

**Важно:** Требуется хранить каждую колоду в **кольцевом буфере** размером ровно на  $(2N + 1)$  элементов. Память под кольцевые буферы выделяйте динамически.

## Пример

input.txt	output.txt
5 0	5 1 2 3 4 5 5 -1 -2 -3 -4 -5
5 10 180FA45DB2	6 -1 2 3 4 5 -5 4 1 -3 -4 -2
3 20 CCCCCCCCCCCC9999999999	6 -1 -2 -3 1 2 3 0

## Комментарий

По второму примеру можно заметить, что команды 0, F, A и 5 никогда ничего не изменяют.

## Задача 5. Арифметические выражения

Источник:	основная
Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	разумное
Ограничение по памяти:	разумное

В данной задаче предлагается вычислить заданное арифметическое выражение. Выражение состоит из целочисленных констант, над которыми выполняются операции сложения, вычитания, умножения и деления. Части выражения могут быть заключены в скобки. Унарный минус разрешается, если непосредственно до него нет знака другой операции. Между токенами может быть любое количество пробелов, а может и не быть пробелов.

Вычислять выражение нужно в вещественных числах. Гарантируется, что не будет деления на ноль и проблем с точностью, и что все целочисленные константы длиной не больше четырёх цифр.

### Формат входных данных

В единственной строке дано выражение, которое нужно вычислить. Гарантируется, что выражение корректно. Длина выражения не превышает 500 000 символов.

### Формат выходных данных

Нужно вывести одно вещественное число — значение выражения. Рекомендуется вывести число с максимальной точностью.

Абсолютная или относительная ошибка вашего ответа **не** должна превышать  $10^{-12}$ .

### Пример

input.txt
-5+ 4 *7- 3* 2 + 17/3- 5 + (3 - 5 + 3) * 2
output.txt
19.666666666666666785090
input.txt
((((3)) + ((-5))))
output.txt
-2.0000000000000000000000