# Задача 1. Простые до N

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

В первой строке содержится целое число $N$ ($2 \leqslant N \leqslant 5\,000$). Нужно вывести все простые числа в диапазоне от 1 до $N$ включительно, по одному числу в строке.

## Пример

| input.txt | output.txt |
|---|---|
| 10 | 2<br>3<br>5<br>7 |

# Задача 2. День недели

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

В единственной строке файла записано три буквы, обозначающие день недели на английском языке. Требуется вывести номер этого дня недели.

В файле нет пробелов. Первая буква заглавная, остальные две маленькие. Гарантируется, что с трёх записанных букв начинается название дня недели на английском языке.

## Пример

| input.txt | output.txt |
|---|---|
| Wed | 3 |

## Пояснение к примеру

Среда в английском называется `Wednesday`.

# Задача 3. Посчитать знаки

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

Требуется определить, какая доля чисел в последовательности отрицательная, какая доля равна нулю, и какая доля положительная.

## Формат входных данных

В первой строке содержится целое число $N$ — количество элементов последовательности ($1 \leqslant N \leqslant 10\,000$). Во второй строке записано $N$ целых чисел через пробел — сама последовательность. Все элементы последовательности по абсолютной величине не превышают 100.

## Формат выходных данных

Нужно вывести три вещественных числа. Первое показывает, какая доля чисел отрицательная. Второе — какая доля чисел равна нулю. И последнее — какая доля положительных чисел.

Каждое выведенное число должно отличаться от своего правильного значения **не** более чем на $10^{-5}$.

## Пример

| input.txt | output.txt |
|---|---|
| 7<br>1 3 1 -2 -1 0 1 | 0.28571 0.14285 0.57142 |

## Пояснение к примеру

В последовательности $2/7$ чисел отрицательны, $1/7$ чисел равна нулю, и $4/7$ чисел положительны.

## Комментарий

Рекомендуется использовать тип `double` для хранения вещественных чисел, а выводить их с помощью формата `"%0.5lf"`, например:

```
double answer = 0.123456789;
printf("%0.5lf", answer);
```

# Задача 4. Прямоугольники

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

Дано три прямоугольника A, B и C, заданных длинами своих сторон. Нужно определить, можно ли расположить их на плоскости так, чтобы выполнялось два условия:

1. все стороны прямоугольников параллельны/перпендикулярны друг другу;
2. прямоугольник A содержит внутри себя прямоугольники B и C (касания сторонами разрешены);

Поворачивать прямоугольники разрешается.

## Формат входных данных

В первой строке содержится шесть целых положительных чисел, записанных через пробел: $a_1$, $a_2$, $b_1$, $b_2$, $c_1$, $c_2$. Длины сторон прямоугольника A равны $a_1$ и $a_2$, прямоугольника B — $b_1$ и $b_2$, а прямоугольника C — $c_1$ и $c_2$. Все числа не превышают 100.

## Формат выходных данных

Нужно вывести слово `YES`, если расположить прямоугольники требуемым образом можно, и `NO` в противном случае.

## Пример

| input.txt | output.txt |
|---|---|
| 5 5 2 5 3 3 | YES |
| 5 5 5 2 4 4 | NO |

## Пояснение к примеру

В первом примере прямоугольник B размерами $2 \times 5$ можно поместить сбоку прямоугольника A, тогда остаётся пустое пространство размера $3 \times 5$, и туда влезает прямоугольник C (размера $3 \times 3$).

Во втором примере площадь A равна 25, а площади B и C равны 10 и 16 соответственно, поэтому искомое вложение заведомо невозможно.

# Задача 5. Слова

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

В единственной строке файла записан набор слов, разделённых символами точки. Каждое слово состоит из букв латинского алфавита. Между словами находится один или несколько символов точки. До первого слова и после последнего точки могут быть, а могут не быть. В строке может быть записано всего одно слово, а может и вовсе не быть слов. Длина заданной строки находится в диапазоне от 1 до 10 000.

Требуется вывести одно целое число — количество слов в строке.

## Пример

| input.txt | output.txt |
|---|---|
| `..ko..Privet.kreved....ko...` | 4 |

## Пояснение к примеру

В примере записано четыре слова: `ko`, `Privet`, `kreved` и `ko`.

## Комментарий

В данной задаче нужно читать символы из входного файла по одному, сохраняя их в переменную типа `char`. Примерно так:

```
char symbol;
scanf("%c", &symbol);
```

Гарантируется, что после строки имеется символ перевода строки `'\n'`.

# Задача 6. Биты и байты

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

Дана последовательность из $N$ битов, каждый бит имеет значение 0 или 1. Нужно разделить эту последовательность на байты, по 8 битов в каждом (в последний байт может попасть меньше битов). После этого нужно распечатать значения всех полученных байтов в привычной людям десятичной системе исчисления.

Биты внутри байта записываются в привычном современным компьютерам порядке little-endian: сначала идут младшие биты, потом старшие.

## Формат входных данных

В первой строке входного файла записано одно целое число $N$ — количество битов в последовательности ($1 \leqslant N \leqslant 100\,000$). Во второй строке записано ровно $N$ символов `0` или `1`: значения битов последовательности.

**Внимание:** после второй строки файла символ перевода строки может быть, а может **не** быть.

## Формат выходных данных

Выведите в одну строку через пробел десятичные значения полученных байтов.

## Пример

| input.txt | output.txt |
|---|---|
| 34<br>0101000011111111000000011100101011 | 10 255 128 83 3 |

## Пояснение к примеру

Разделим в примере биты на группы (байты). Обратите внимание, что в последнюю группу попадает только 2 бита. Далее преобразуем байты в десятичный вид:

- `01010000` $= 2 + 8 = 10$
- `11111111` $= 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$
- `00000001` $= 128 = 128$
- `11001010` $= 1 + 2 + 16 + 64 = 83$
- `11` $= 1 + 2 = 3$

# Задача 7. Арифметические прогрессии

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

В файле дано три целых числа $L$, $R$ и $K$ ($1 \leqslant L < R \leqslant 5\,000$, $2 \leqslant K \leqslant 1\,000$). Требуется найти количество возрастающих арифметических прогрессий, у которых ровно $K$ элементов лежит в диапазоне от $L$ до $R$ включительно.

В данной задаче нужно рассматривать только прогрессии, у которых все элементы целые. Считается, что у арифметической прогрессии есть первый элемент, но нет последнего: она бесконечная. Если две прогрессии начинаются с разного элемента, то их нужно считать различными.

## Пример

| input.txt | output.txt |
|---|---|
| 10 20 3 | 10 |

## Пояснение к примеру

Все искомые арифметическии прогрессии для примера:

- **10 14 18** 22 26 ...
- **10 15 20** 25 30 ...
- **11 15 19** 23 27 ...
- **12 15 18** 21 24 ...
- **12 16 20** 24 28 ...
- **13 16 19** 22 25 ...
- **14 17 20** 23 26 ...
- **15 17 19** 21 23 ...
- **16 18 20** 22 24 ...
- **18 19 20** 21 22 ...

# Задача 8. Дата

| | |
|---|---|
| Источник: | повышенной сложности |
| Имя входного файла: | `input.txt` |
| Имя выходного файла: | `output.txt` |
| Ограничение по времени: | 1 секунда |
| Ограничение по памяти: | разумное |

Дано четыре целых числа $D$, $M$, $Y$ и $K$. Первые три задают корректную дату, считая что $D$ — это номер дня в месяце, $M$ — номер месяца в году, и $Y$ — номер года от рождества Христова. Нужно найти, какая дата будет через $K$ дней после этой, и вывести её в таком же формате (т.е. день месяц год).

Ограничения: $Y \leqslant 10\,000$, $K \leqslant 1\,000\,000$

В данной задаче исчисление ведётся по *григорианскому календарю*. То есть год $N$ считается високосным, если верно одно из:

- $N$ делится на 400
- $N$ делится на 4, но **не** делится на 100

## Пример

| input.txt | output.txt |
|---|---|
| 15 1 2018 70 | 26 3 2018 |
| 15 8 2018 3650 | 12 8 2028 |

## Пояснение к примеру

В первом примере имеется дата: 15 января 2018 года. Поскольку в январе 31 дней, а в фервале 2018 года 28 дней, то через 70 дней получается дата: 26 марта 2018 года.

Во втором примере дана начальная дата 15 августа 2018 года, требуется найти дату через 3650 дней. Если бы в каждом году было 365 дней, то получилось бы 15 августа 2028 года. Однако есть високосные годы 2020-ый, 2024-ый и 2028-ой, поэтому получается 12 августа 2028 года.

# Задача 9. Комментарии

| | |
|---|---|
| Источник: | повышенной сложности* |
| Имя входного файла: | `input.txt` |
| Имя выходного файла: | `output.txt` |
| Ограничение по времени: | 1 секунда |
| Ограничение по памяти: | **специальное** |

Дан текст, нужно удалить из него комментарии согласно правилам языка C (точнее C99). Есть два вида комментариев:

- *Блочный комментарий*: начинается с `/*`, заканчивается `*/`, может занимать несколько строк.
- *Строчный комментарий*: начинается с `//` и заканчивается символом перевода строки.

При удалении комментария все символы перевода строки остаются в файле, даже если они находятся внутри блочного комментария. В конце текста может остаться открытый комментарий. В данной задаче **все пробелы и переводы строк в выходном файле должны быть выведены точно**.

Текст непустой, имеет длину до миллиона символов. Объём используемой вашей программой памяти должен быть много меньше мегабайта. В частности, сохранять в памяти программы всё содержимое входного файла **нельзя**.

В тексте могут быть следующие символы:

- маленькие и большие латинские буквы,
- цифры,
- пробелы и переводы строк,
- символы `/` и `*`,
- дополнительные символы: круглые и фигурные скобки, запятая и точка с запятой, знаки плюс и минус.

## Пример

| input.txt | output.txt |
|---|---|
| `/*C-style comments can contain`<br>`multiple lines*/ /*or just one*/`<br>`// C++-style comment lines`<br>`int main() {`<br>`  // The below code wont be run`<br>`  //return 1;`<br>`  return 0; //this will be run`<br>`}` | `int main() {`<br><br><br>`  return 0;`<br>`}` |

## Комментарий

Рекомендуется читать текст по символам. Для проверки, закончились ли символы в файле, можно сравнивать возвращаемое значение `scanf` с единицей:

```
while (1) {
    if (scanf("%c", &curr) != 1)
        break;
    ... //читаем и обрабатываем очередной символ
}
```