# Задача 1. Сумма от 1 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 входного файла содержится целое число $N$ ($1 \leqslant N \leqslant 100$).

В выходной файл нужно вывести сумму всех целых чисел, лежащих в диапазоне от 1 до $N$ включительно.

## Пример

| input.txt | output.txt |
|---|---|
| 5 | 15 |

## Пояснение к примеру

В примере получается $1 + 2 + 3 + 4 + 5 = 15$

# Задача 2. Сумма чётных

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

Дана последовательность целых чисел, требуется посчитать сумму всех её чётных элементов.

## Формат входных данных

В первой строке входного файла содержится целое число $N$ — количество элементов последовательности ($1 \leqslant N \leqslant 100$).

Во второй строке записано $N$ целых чисел через пробел — сама последовательность. Все элементы последовательности по абсолютной величине не превышают 100.

## Формат выходных данных

В выходной файл необходимо вывести одно целое число: сумму всех чётных элементов последовательности.

## Пример

| input.txt | output.txt |
|---|---|
| 8<br>2 3 7 6 8 3 1 2 | 18 |

## Пояснение к примеру

В примере получается $2 + 6 + 8 + 2 = 18$

# Задача 3. Проверка на простоту

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

По определению, простым числом называется целое число, большее 1, которое делится нацело только на себя и на 1.

Требуется написать функцию, определяющую, является ли заданное число простым, и использовать ее для решения данной задачи.

Функция должна иметь сигнатуру:

```
int prime ( int n );
```

Здесь `n` — положительное целое число, функция должна выдать 1, если `n` – простое число, и 0 – иначе.

## Формат входных данных

Во входном файле записано целое число $N$ ($1 \leqslant N \leqslant 10\,000$).

## Формат выходных данных

В выходной файл нужно вывести слово `YES`, если $N$ является простым, и слово `NO` в противном случае.

## Пример

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

## Пояснение к примеру

Число 5 простое, а число $4 = 2 \times 2$ составное.

# Задача 4. Простые числа

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

Задано натуральное число $N$. Необходимо найти все простые числа, не превосходящие заданного $N$, и вывести их в порядке возрастания.

## Формат входных данных

Во входном файле записано одно натуральное число $N$ ($2 \leqslant N \leqslant 10^6$) .

## Формат выходных данных

В выходной файл необходимо вывести в порядке возрастания через пробел все простые числа, не превосходящие $N$.

## Пример

| input.txt | output.txt |
|---|---|
| 23 | 2 3 5 7 11 13 17 19 23 |

## Замечания

Для проверки числа на простоту использовать функцию `prime`, описанную в предыдущей задаче.

# Задача 5. Обращение числа

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

Определить число, получаемое выписыванием в обратном порядке цифр заданного натурального числа.

## Формат входных данных

Во входном файле записано число $N$ ($1 \leqslant N < 10^8$).

## Формат выходных данных

В выходной файл необходимо вывести требуемое натуральное число.Ведущие нули не писать.

## Пример

| input.txt | output.txt |
|---|---|
| 12345 | 54321 |

## Замечания

В данной задаче необходимо сначала получить целое число из заданного $N$, а затем его вывести в выходной файл.

Для этого нужно написать функцию со следующей сигнатурой:

```
int rev_int ( int N );
```

Здесь `N` — положительное целое число, функция должна выдать искомое целое число.

# Задача 6. Посчитать знаки

| | |
|---|---|
| Источник: | основная |
| 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 входного файла 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

В выходной файл нужно вывести через пробел 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 7. Минимум и максимум

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

Дана последовательность целых чисел, требуется найти в ней минимальный и максимальный элементы.

## Формат входных данных

В первой строке входного файла содержится целое число $N$ — количество элементов последовательности ($1 \leqslant N \leqslant 20$).

Во второй строке записано $N$ целых чисел через пробел — сама последовательность. Все элементы последовательности по абсолютной величине не превышают 10 000.

## Формат выходных данных

Требуется вывести четыре целых числа: минимальный элемент последовательности, номер этого минимального элемента, максимальный элемент последовательности и номер максимального элемента. Если минимальных/максимальных элементов несколько, требуется вывести номер первого из них.

В данной задаче элементы нумеруются по порядку, начиная с единицы.

## Пример

| input.txt | output.txt |
|---|---|
| 5<br>1 7 3 4 9 | 1 1 9 5 |

## Пояснение к примеру

В примере дана последовательность из пяти чисел: 1 7 3 4 9

Минимальное число равно 1 и стоит на первой позиции, а максимальное число равно 9 и стоит на последней (пятой) позиции.

# Задача 8. Близкий к целому

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

Задана последовательность вещественных чисел $a_1$, $a_2$, ... $a_n$.

Требуется вычислить порядковый номер того из них, которое наиболее близко к ближайшему целому числу. Если таких чисел несколько, то вывести первое из них.

## Формат входных данных

В первой строке входного файла записано число $n$ ($1 \leqslant n \leqslant 1000$).

Во второй строке через пробел записаны вещественные числа $a_1$, $a_2$, ... $a_n$, каждое из которых по модулю не превосходит 1000.

## Формат выходных данных

В выходной файл необходимо вывести одно целое число — порядковый номер числа, наиболее близкого к целому числу. Если таких элементов несколько, то вывести первый из них. Нумерация чисел начинается с 1.

## Пример

| input.txt | output.txt |
|---|---|
| 4<br>1.7 81.95 -19.05 135.1 | 2 |

# Задача 9. Сумма ряда

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

Не используя стандартные функции, вычислить с точностью eps $> 0$:
$y = arctg(x) = x - x^3/3 + x^5/5 - \ ... \ + (-1)^n x^{2n+1}/(2n+1) + \ ...$

Считать, что требуемая точность достигнута, если очередное слагаемое по модулю меньше *eps*, – все последующие слагаемые можно не учитывать.

## Формат входных данных

В первой строке входного файла записано число $x$ с точностью до двух знаков после десятичной точки ($-1 < x < 1$).

Во второй строке записано число *eps* с точностью до пяти знаков после десятичной точки ($0 < eps < 1$).

## Формат выходных данных

В выходной файл необходимо вывести единственное вещественное число – сумму ряда с точностью до 5 знаков после десятичной точки.

## Пример

| input.txt | output.txt |
|---|---|
| 0.1<br>0.001 | 0.09967 |

# Задача 10. Цифра

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

Найти $k$-ю цифру последовательности 1234567891011121314..., в которой выписаны подряд все натуральные числа.

## Формат входных данных

Во входном файле записано единственное натуральное число $k$ ($0 < k \leqslant 10^8$).

## Формат выходных данных

В выходной файл необходимо вывести $k$-ю цифру заданной последовательности.

## Пример

| input.txt | output.txt |
|---|---|
| 11 | 0 |

# Задача 11. Количество боксов

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

Во входном файле содержится целое число $N$ ($1 \leqslant N \leqslant 10^9$).

Нужно найти количество прямоугольных параллелепипедов с целочисленными сторонами, объём которых не превышает $N$.

Параллелепипеды, которые можно перевести друг в друга с помощью поворота, считаются одинаковыми.

**Замечание:** Ответ к этой задаче может быть настолько большим, что не войдёт в переменную типа `int`. Используйте 64-битный тип следующим образом:

```
long long answer;
answer = 1000000000;
answer = answer * 1000000000;
printf("%lld", answer);
```

## Пример

| input.txt | output.txt |
|---|---|
| 10 | 16 |
| 1000000000 | 39218340164 |

## Пояснение к примеру

Вот все возможные тройки размеров из первого примера:

- 1 1 1
- 1 1 2
- 1 1 3
- 1 1 4
- 1 1 5
- 1 1 6
- 1 1 7
- 1 1 8
- 1 1 9
- 1 1 10
- 1 2 2
- 1 2 3
- 1 2 4
- 1 2 5
- 1 3 3
- 2 2 2