Задача 1. Минимальная дата

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

Требуется реализовать функцию для поиска самой ранней даты среди заданного массива дат. Даты включают в себя не только день, но и точное время. При помощи реализованной функции нужно решить тестовую задачу.

Каждая дата должна представляться структурой:

```
typedef struct DateTime_s {
    int year, month, day;
    int hours, minutes, seconds;
} DateTime;
```

Функция для поиска самой ранней (минимальной) даты должна иметь сигнатуру:

```
DateTime min(const DateTime *arr, int cnt);
```

Здесь `arr` — указатель на первый элемент массива дат, а `cnt` — длина массива.

Формат входных данных

В первой строке содержится целое число N — количество дат в файле ($2 \leqslant N \leqslant 50\,000$). В каждой из следующих N строк описана одна дата в виде шести целых чисел: `year`, `month`, `day`, `hours`, `minutes`, `seconds`. Гарантируется, что все даты корректны, а год лежит в пределах от 1 до 5 000 включительно.

Формат выходных данных

Нужно вывести самую раннюю дату среди записанных в файле дат в том же формате, в котором даты записываются во входных данных.

Пример

input.txt	output.txt
5 2018 8 12 23 44 13 2018 9 1 9 0 0 2019 1 1 0 0 0 2018 2 13 13 1 7 2018 8 26 8 20 11	2018 2 13 13 1 7

Задача 2. Биномиальные коэффициенты

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

Биномиальный коэффициент C_n^k — это количество битовых массивов длины n, в которых ровно k битов единичные. Подробнее о биномиальных коэффициентах можно прочитать, например, в википедии.

Нужно вычислить все биномиальные коэффициенты для $n \leqslant 1\,000$ при помощи треугольника Паскаля. Подробнее о нём можно прочитать, опять же, в википедии. Треугольник Паскаля позволяет вычислять все биномиальные коэффициенты в порядке увеличения n, т.к.:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k \quad \text{при } 0 < k < n$$

После того, как все коэффициенты вычислены, нужно прочитать набор пар n и k из входного файла и выдать для каждой пары соответвующий коэффициент C_n^k.

Формат входных данных

В первой строке содержится целое число Q — количество запросов в файле ($1 \leqslant Q \leqslant 10\,000$). В каждой из следующих Q строк содержится по два целых числа n и k, для которых нужно распечатать коэффициент ($0 \leqslant k \leqslant n \leqslant 1\,000$).

Формат выходных данных

Нужно вывести Q вещественных чисел, по одному в строке — биномиальные коэффициенты для запросов из входном файле.

Внимание: хоть биномиальные коэффициенты и целые, они могут быть очень большими. Поэтому вычисляйте их как вещественные числа с использованием типа `double`, и распечатывайте при помощи формата `"%0.10g"` !

Пример

input.txt	output.txt
8 4 0 4 1 4 2 4 3 4 4 10 5 100 50 1000 500	1 4 6 4 1 252 1.008913445e+29 2.702882409e+299

Пояснение к примеру

Первые пять запросов распечатывают коэффициенты для $n = 4$. Последний запрос распечатывает самый большой коэффициент, который может быть запрошен в данной задаче.

Комментарий

Указанное выше свойство треугольника Паскаля можно легко доказать, если заметить, что коэффициенты при степенях $(1 + x)^n$ можно выразить через коэффициенты в произведении $(1 + x)^{n-1} \cdot (1 + x)$.

Задача 3. Имена и возрасты

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

Есть набор рисунков с подписями автором в стиле “Зоя 17 лет”. Нужно реализовать функцию для сбора статистики по именам и возрастам авторов.

Каждая подпись должна представляться структурой:

```
typedef struct Label_s {
    char name[16];   //имя автора (заканчивается нулём)
    int age;         //возраст автора (сколько лет)
} Label;
```

Статистика имён должна представляться структурой:

```
typedef struct NameStats_s {
    int cntTotal;    //сколько всего подписей
    int cntLong;     //сколько подписей с именами длиннее 10 букв
} NameStats;
```

Статистика возрастов должна представляться структурой:

```
typedef struct AgeStats_s {
    int cntTotal;    //сколько всего подписей
    int cntAdults;   //сколько подписей взрослых (хотя бы 18 лет)
    int cntKids;     //сколько подписей детей (меньше 14 лет)
} AgeStats;
```

Функция для вычисления статистик должна иметь сигнатуру:

```
void calcStats(const Label *arr, int cnt, NameStats *oNames, AgeStats *oAges);
```

Здесь `oNames` и `oAges` — адреса структур, в которые нужно записать результат.

Формат входных данных

В первой строке содержится целое число N — количество подписей в файле ($1 \leqslant N \leqslant 1\,000$). Каждая подпись записана в виде “`[имя] [возраст] let`”.

Все имена не длиннее 15 символов, возрасты целые, положительные, не больше 5 000.

Формат выходных данных

Нужно вывести статистики `NameStats` и `AgeStats` в формате, приведённом в примере.

Пример

input.txt	output.txt
7 Zoya 17 let Kirill 5 let Ivan 1 let Vasiliy 15 let Tutankhamun 3360 let Innokentiy 21 let Dozdraperma 70 let	names: total = 7 names: long = 2 ages: total = 7 ages: adult = 3 ages: kid = 2

Задача 4. Простые числа

Источник:	основная
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	2 секунды*
Ограничение по памяти:	разумное

Найдите все простые числа меньшие N.

Формат входных данных

В первой строке содержатся два целых числа: N — диапазон, в котором нужно найти все простые числа и Q — количество запросов в файле ($10 \leqslant N \leqslant 20\,000\,000$, $1 \leqslant Q \leqslant 200\,000$).

В каждой из следующих Q строк содержится по одному целому числу X, для которого нужно вывести, простое оно или нет ($0 \leqslant X < n$).

Формат выходных данных

Нужно вывести Q ответов, по одному в каждой строке. Каждый ответ — это само число, указанное в запросе, и слово `prime` или `not` в зависимости то того, является число простым или нет.

Пример

input.txt	output.txt
10 8 0 1 2 4 7 5 6 9	0 not 1 not 2 prime 4 not 7 prime 5 prime 6 not 9 not

Задача 5. Числа Фибоначчи

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

Последовательность чисел Фибоначчи определяется следующим образом. Первое и второе числа Фибоначчи равны единице, а каждое следующее число Фибоначчи равно сумме двух предыщущих. Вот первые числа Фибоначчи: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Во входном файле задано одно целое число N в диапазоне $1 \leqslant N \leqslant 2\,000$. Нужно вычислить и распечатать N-ое число Фибоначчи.

Поскольку это число может быть довольно большим (более 400 цифр), то вам необходимо собственноручно реализовать десятичную арифметику. Для этого рекомендуется использовать следующую структуру длинного числа:

```
typedef struct LongNum_s {
    int len;          //сколько цифр в числе
    int arr[500];     //массив десятичных цифр числа
} LongNum;
```

Далее вам следует реализовать алгоритм сложения длинных чисел “в столбик”, как учили в школе, а также написать функцию распечатывания длинного числа. Тогда вы сможете вычислить нужное число Фибоначчи простым циклом по N.

Пример

input.txt	output.txt
12	144

Комментарий

Рекомендуется хранить цифры в массиве `arr` в обратном принятому у людей порядке: `arr[0]` — это единицы, `arr[1]` — десятки, а `arr[len-1]` — это ведущая цифра. Кстати, такой порядок “по-умному” называется little-endian.

Задача 6. Факториал

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

Факториал от N (обозначается как $N!$) — это произведение всех целых чисел от 1 до N включительно.

Во входном файле задано одно целое число N в диапазоне $1 \leqslant N \leqslant 1\,000$. Нужно вычислить и распечатать факториал от N.

На этот раз вам понадобится обеспечить 3 000 цифр в структуре длинного числа, чтобы влез результат. Для вычисления ответа можно перебирать все k по порядку от 1 до N, и каждый раз домножать текущее длинное число на него.

Здесь нужно также реализовать умножение “в столбик”, однако можно полагать множитель одной цифрой (даже когда он больше десяти). В некотором смысле, эта операция называется умножением длинного числа (заданного массивом цифр) на короткое (заданное одним числом), и сама по себе она реализуется без вложенных циклов.

Пример

input.txt	output.txt
10	3628800

Задача 7. Длинное умножение

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

Во входном файле задано два целых положительных числа, одно в первой строке, второе — во второй строке. Нужно вычислить произведение этих двух чисел и вывести его в выходной файл.

Количество десятичных цифр в каждом входном числе может достигать 1 000. Поэтому требуется реализовать умножение двух длинных чисел “столбиком”.

Пример

input.txt	output.txt
1991 30457	60639887

Задача 8. Числа со знаком

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

Дана последовательность из N целых чисел. Некоторые из них заданы явно (как число в виде десятичной записи), а некоторые определяются как сумма, разность или произведение двух более ранних чисел последовательности. Нужно найти и вывести все N чисел, выполнив заданные арифметические операции.

Формат входных данных

В первой строке задано целое число N — количество чисел в последовательности ($1 \leqslant N \leqslant 1\,000$). В каждой из следующих N строк задано одно число. Числа нумеруются в порядке задания, начиная с нуля.

Описание k-ого числа начинается с цифры t, определяющей способ задания. Если $t = 0$, то дальше через пробел записано само число в явном виде.

Если $t = 1$, то данное число определяется как сумма, если $t = 2$ — то как разность, а если $t = 3$ — то как произведение. В любом из этих трёх случаев далее через пробел указано два целых числа a и b — номера членов последовательности, над которыми нужно выполнить арифметическую операцию, чтобы получить k-ое число ($0 \leqslant a, b < k$).

Гарантируется, что каждое число искомой последовательности по модулю меньше 10^{100}.

Формат выходных данных

Требуется вывести все N чисел последовательности, по одному целому числу в строке.

Пример

input.txt	output.txt
9 0 12 0 15 3 0 1 0 135 2 3 2 2 0 1 3 5 4 2 0 0 1 4 6	12 15 180 135 -45 -3 135 0 90
5 0 100000000 3 0 0 3 1 1 0 10000000000000000000000000000000 2 3 2	100000000 10000000000000000 100000000000000000000000000000000 10000000000000000000000000000000 -990000000000000000000000000000000

Комментарий

Крайне рекомендуется использовать дополнительный код для задания отрицательных чисел, и соответственно использовать длинные числа фиксированной длины.

Задача 9. Длинное деление

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 задано два целых положительных числа, одно в первой строке, второе — во второй строке. Нужно вычислить частное этих двух чисел (округлённое вниз) и вывести его в выходной файл.

Количество десятичных цифр в каждом входном числе может достигать 1 000. Поэтому требуется реализовать деление двух длинных чисел “столбиком”.

Пример

input.txt	output.txt
1023405633 84037	12178
10000000000000000000000000 7	14285714285714285714285714

Задача 10. Длинное деление +

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 задано два целых положительных числа, одно в первой строке, второе — во второй строке. Нужно вычислить частное этих двух чисел (округлённое вниз) и вывести его в выходной файл.

Количество десятичных цифр в каждом входном числе может достигать 25 000.

Для решения этой задачи необходимо хранить не по одной десятичной цифре в каждом элементе массива, а хотя бы по четыре цифры, в некотором смысле храня длинное число в 10 000-ичной системе исчисления. В процессе деления столбиком нужно на каждом шаге подбирать цифру ответа, и делать это перебором 10 000 возможных вариантов уже неприемлемо. Определение цифры следует осуществлять бинарным поиском.

Пример

input.txt	output.txt
1023405633 84037	12178
10000000000000000000000000 7	142857142857142857142857

Задача 11. Длинное деление ++

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

Во входном файле задано два целых положительных числа, одно в первой строке, второе — во второй строке. Нужно вычислить частное этих двух чисел (округлённое вниз) и вывести его в выходной файл.

Обозначим количество десятичных цифр в первом числе через N, а во втором — через M. Тогда верны следующие ограничения:

- $N, M \leqslant 10^6$
- $M \cdot (N - M) \leqslant 2.5 \cdot 10^9$

Для решения этой задачи необходимо:

1. Хранить в каждом элементе массива по 9 десятичных цифр (для промежуточных результатов нужны 64-битные целые числа).
2. Находить каждую цифру частного за $O(1)$ попыток против $O(log_2 B)$ попыток для бинарного поиска (здесь $B = 10^9$).
3. Использовать алгоритм с общей асимптотикой $O(M(N - M))$, а не просто любое квадратичное решение.

Пример

input.txt	output.txt
1023405633 84037	12178
100000000000000000000000000 7	14285714285714285714285714