Задача 1. 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 2. Слова из трех букв

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

Ваши друзья пытаются заполнить кроссворд, нужно подобрать слово из трех букв, так, чтобы на втором месте стояла буква `A`. Они неоднократно предлагают друг другу слова, надеясь, что они подходят. Кто-то предлагает написать программу, которая может быстро определить, соответствует ли данное слово приведенному выше описанию.

Ваша задача — написать такую программу.

Формат входных данных

Во входном файле записано одно слово, состоящее из прописных латинских букв. Его длина не превышает 12 символов. После слова записан символ «точка».

Формат выходных данных

В выходной файл необходимо вывести текст, который обозначает, подходит или нет введенное слово. Если слово состоит из трех букв, и на втором месте стоит буква `A`, то нужно вывести `FITS`. В противном случае, ваша программа должна вывести `DOES NOT FIT`.

Примеры

input.txt	output.txt
CAT.	FITS
FATE.	DOES NOT FIT
TEN.	DOES NOT FIT

Задача 3. Что бы ты ни сделал, я могу сделать лучше

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

У вас есть друг, который слишком любит соревноваться, всегда старается вас «переплюнуть». Если вы говорите, что ваша машина быстрая, то ваш товарищ обязательно скажет, что его машина еще быстрее. Если вы скажете, что ваша машина быстрее, то он заявит, что его – самая быстрая. После нескольких таких разговоров вы понимаете, что всегда можно предугадать, что ваш товарищ скажет в следующий момент.

Чтобы продемонстрировать, как это все надоедает, вы решили написать программу, которая аккуратно предсказывает ответы вашего друга. В частности, для любого английского прилагательного, ваша программа будет возвращать его же в сравнительной форме путем добавления `er` к нему. Заметим, что если прилагательное уже заканчивается на `e`, вы должны только добавить букву `r`. Если же программе на вход дано прилагательное уже в сравнительной форме, то программа должна вернуть это прилагательное в превосходной форме путем простой замены `er` на `est`.

Формат входных данных

Во входном файле записано одно слово, после которого идет символ точка. Слово представляет собой прилагательное или сравнительную форму прилагательного. Слово состоит только из строчных латинских букв. Его длина не превосходит 30

Формат выходных данных

В выходной файл необходимо вывести преобразованное прилагательное в сравнительной или в превосходной форме, как описано в условии.

Примеры

input.txt	output.txt
`warm.`	`warmer`
`smaller.`	`smallest`
`rare.`	`rarer`

Задача 4. Делимость на 15

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

На вход подается число. Нужно определить, делится ли оно на 15.

Формат входных данных

Входной файл состоит из последовательности десятичных цифр, за которыми следует символ «точка». Длина заданного числа не превышает 500.

Формат выходных данных

Если введенное число делится нацело на 15, то в выходной файл необходимо вывести слово `YES`. В противном случае необходимо вывести слово `NO`.

Примеры

input.txt	output.txt
12345.	YES
67.	NO

Задача 5. Количество слов

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. Буквы алфавита

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

Вам дана строка, состоящая из строчных букв латинского алфавита. Все буквы в ней различны.

Требуется переставить буквы данной строки так, чтобы получившаяся строка была лексикографически больше исходной.

Поскольку в данной постановке задача слишком простая и может иметь не единственное решение, то вам требуется среди всех таких строк выбрать лексикографически минимальную.

Строка s, состоящая из символов s_0, s_1, …, s_n, считается *лексикографически меньше* строки t, состоящей из символов t_0, t_1, …, t_n, если сущесвует индекс k такой, что:

- $s_i = t_i$ для всех $i = 0, 1, \ldots, k-1$;
- $s_k < t_k$.

Иными словами, лексикографическое сравнение строк — это привычное нам сравнение слов «по алфавиту», когда мы находим первую букву, в которой две строки различаются, и на основании этой буквы делалем вывод о том, какое из слов «меньше». Лексикографическое сравнение окружает нас повсюду: его можно найти в порядке людей в списках групп, в порядке номеров в телефонной книге, и т.д.

Формат входных данных

В первой строке входного файла записано целое число N — количество символов в строке ($2 \leqslant N \leqslant 26$).

Во второй строке через пробел записано N строчных букв латинского алфавита. Гарантируется, что все буквы различны.

Формат выходных данных

В выходной файл необходимо вывести через пробел символы требуемой строки.

Гарантируется, что требуемая перестановка существует.

Примеры

input.txt	output.txt
5 a b c d e	a b c e d
3 q z w	w q z

Задача 7. Сумма чисел

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

В заданном тексте нужно посчитать сумму входящих в него десятичных целых чисел.

Формат входных данных

Во входном файле записана непустая последовательность символов, среди которых могут встречаться цифры. Эта последовательность заканчивается символом «точка». Длина последовательности не превосходит 10^4. Числа, которые в ней встречаются, имеют не более 5 знаков.

Формат выходных данных

В выходной файл необходимо вывести целое число – сумму чисел, которые встречаются в заданной последовательности.

Пример

input.txt	output.txt
`This is example: three (3) + 15 is equal to 18.`	`36`

Задача 8. 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`
`multiple lines*/ /*or just one*/`
`// C++-style comment lines`
`int main() {`
`  // The below code wont be run`
`  //return 1;`
`  return 0; //this will be run`
`}`
	`int main() {`
	
	`  return 0;`
	`}`

Комментарий

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

```
while ( scanf("%c", &curr) == 1) {

    ... //читаем и обрабатываем очередной символ
}
```

Задача 9. 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