кафедра Информационных технологий и программной инженерии
Беляков Андрей Юрьевич
VK: permcube
Telegram: @AndreyPerm

Алгоритмы и структуры данных

Контрольная работа

(для заочной формы обучения)

Все материалы и особенности учебной дисциплины обсуждались на очном занятии и были опубликованы тут:

https://github.com/permCoding/algorithms-and-data-structures/tree/master/for-external-student

Но, так как Stepik стал платным, то теперь задачи **следует выполнять без «степика»** и передать мне на проверку либо через repl.it, либо через публикацию папки с решениями в виде репозитория на git`е. Можно ваши решения передать мне просто в виде архива zip, rar. Все остальные правила и требования остаются без изменений… Не надо оформлять бумажный вариант контрольной работы, требуется сделать **только электронный вариант** – просто сами программы…

Сами задачи, которые нужно было решить на Stepik`е, а это «Входной контроль» и раздел «Разминка», теперь не доступны, так как доступ сейчас стал платным. По этой причине тексты задач по программированию я продублирую тут ниже.

Для решения задач можно использовать любой язык программирования из списка самых популярных процедурных языков: C++, C#, java, Python.

Как правило, для задачи требуется входной поток данных – оформляйте его как стандартный ввод с клавиатуры в консоли. Ответ тоже следует оформлять в виде вывода в консоль.

Например, в Питоне так оформляется ввод одного целого числа:

```
x = int(input())
```

и так можно оформить вывод на экран результата:

```
print(result)
```

Если требуется вывестицелый список, то в задаче будет уточнён формат вывода: через пробел в одной строке или «в столбик».

Как правило, к каждой задаче прилагаются примеры входных (Sample Input) и выходных (Sample Output) потоков данных, они помогут вам разобраться в задаче и понять, как оформлять вывод результата.

Экзамен будет проходить в форме собеседования по вашим решениям этих задач.

Оглавление

Входной контроль

В этом разделе простые задачи для проверки остаточных знаний. Предполагается знание основных конструкций языка программирования.

1.1 Строковый градусник

На вход подаётся температура на улице в целых градусах в диапазоне от -50 до +50.

На выход нужно подать ответ строку:

- `жутко холодно`, если температура ниже -25°C
- `холодно`, если температура выше -26°, но ниже 0°C
- `прохладно`, если температура выше -1°, но ниже 10°C
- `тепло`, если температура выше 9°, но ниже 25°C
- `жара`, если температура выше 24°

Данные следует считывать из стандартного входного потока, например для Питона так:

```
x = int(input())
```

Ответ следует выводить в стандартный поток вывода, например для Питона так:

```
print(result)
```

Sample Input 1:

```
0
```

Sample Output 1:

```
прохладно
```

Sample Input 2:

```
35
```

Sample Output 2:

```
жара
```

Sample Input 3:

```
-15
```

Sample Output 3:

```
холодно
```

1.2 Любое чётное

На вход подаётся список из **n** `[1, 1000]` целых чисел из диапазона `[0, 1000]`, записанных в одну строку через пробел.

На выход ваша программа должна вернуть **любое чётное** число из этого списка.

Гарантируется, что в списке будет не менее одного чётного числа.

Данные следует считывать из стандартного входного потока, например для Питона так:

```
# исходную строку разбиваем по пробелам на список подстрок
line = input()
list_substrings = line.split()
```

Ответ следует выводить в стандартный поток вывода, например для Питона так:

```
print(result)
```

Sample Input 1:

```
0 3 7 2 9 8 8 2 0 0
```

Sample Output 1:

```
0
```

Sample Input 2:

```
7 7 1 4 7 6 4 1 5 4
```

Sample Output 2:

```
4
```

1.3 Минимальное число

Напишите программу, которая в последовательности натуральных чисел определяет **минимальное число, оканчивающееся на 3**.

На вход программа получает:
- в первой строке одно целое число N - количество чисел в последовательности;
- в последующих N строках сами числа последовательности.

В последовательности всегда имеется хотя бы одно число, оканчивающееся на 3. Количество чисел не превышает 1000. Введённые числа не превышают 30 000.

Программа должна вывести одно число – **минимальное число, оканчивающееся на 3**.

Для получения данных сначала считайте первую строку и извлеките число N, затем запустите цикл с N повторами и считайте N строк с данными для программы.

Sample Input 1:

```
4
63
10
531
11
```

Sample Output 1:

```
63
```

Sample Input 2:

```
7
1
11
10
31
33
103
993
```

Sample Output 2:

```
33
```

1.4 Чётные против нечётных

Напишите программу, которая в последовательности натуральных чисел определяет количество чётных и нечётных чисел. Ответ выводится в виде одного слова:
- **equal** - если чисел одинаковое количество,
- **odd** - если нечётных больше,
- **even** - если чётных больше.

На вход программа получает одну строку с некоторым количеством (от 1 до 1000 включительно) целых положительных чисел (значения чисел от 1 до 30000 включительно), записанных через пробел.
На выход программа должна вывести одну строку с ответом - в зависимости от сравнения количества чётных и нечётных.

Sample Input 1:

```
1 2 3 4
```

Sample Output 1:

```
equal
```

Sample Input 2:

```
1 3 5 2
```

Sample Output 2:

```
odd
```

Sample Input 3:

```
1 2 4 2
```

Sample Output 3:

```
even
```

Раздел 2 - РАЗМИНКА включает три темы:
- Арифметика (5 задач),
- Матрицы (4 задачи),
- Строки (9 задач).

Арифметика

2.1.1 Свой-Чужой

Жители племени «Тумба-Юмба» единственные в долине, которые умеют правильно отвечать на один особенный вопрос. По правильному ответу их можно отличить от жителей других племён. Вот этот вопрос: назовите сумму целых чисел от **1** (включительно) до **N** (включительно).

Напишите программу для проверки жителей.

На вход в консоли с клавиатуры вводится одно целое число **N** (**по модулю не более 10^4**).

На выход – на экран выводится искомая сумма.

Sample Input 1:

```
1
```

Sample Output 1:

```
1
```

Sample Input 2:

```
3
```

Sample Output 2:

```
6
```

Sample Input 3:

```
-3
```

Sample Output 3:

```
-5
```

2.1.2 Прямоугольники

Дети племени «Тумба-Юмба» любят играть в логические игры. Однажды вождь племени придумал детям задачу на построение прямоугольников одинаковой площади. Пусть дана площадь прямоугольника, тогда нужно найти количество различных прямоугольников с целочисленными длинами сторон заданной площади. Например, для площади 20 можно построить три различающихся прямоугольника с такой же площадью.

Итак, на вход подаётся одно целое положительное число, не превосходящее 10^9 – это заданная площадь прямоугольника.

На выход нужно вывести на экран искомое количество прямоугольников.

Sample Input 1:

```
20
```

Sample Output 1:

```
3
```

Sample Input 2:

```
10
```

Sample Output 2:

```
2
```

Sample Input 3:

```
9
```

Sample Output 3:

```
2
```

2.1.3 Результаты охоты

Вождь племени «Тумба-Юмба» очень аккуратен в подсчёте успехов на охоте, но он особенным образом записывает результаты. В местных лесах водятся 9 разных видов животных – они обозначаются **цифрами от 1 до 9**. После охоты вождь раскладывает трофеи по видам животных по возрастанию номеров подряд, например, так:

334555

то есть пойманы две «тройки», одна «четвёрка», три «пятёрки», и, затем, так же как мы только что назвали результаты, так и записывает их:

231435

Вождь таким образом шифрует результаты охоты.

Помогите вождю – напишите программу, которая из входной строки будет делать зашифрованную. Длина входной строки не превышает **100** символов.

Sample Input 1:

```
334555
```

Sample Output 1:

```
231435
```

Sample Input 2:

```
4444555667
```

Sample Output 2:

```
44352617
```

Sample Input 3:

```
1
```

Sample Output 3:

```
11
```

2.1.4 Тот-кого-нельзя-называть

Чтобы настроение жителей Солнечного города находилось по «Шкале Счастья» в рамках от Хорошего до Прекрасного достаточно обеспечить такое распределение зарплат, чтобы коротышка с самой низкой зарплатой получал не менее 10% от самой большой зарплаты в городе.

Самый активный житель города – `Тот-кого-нельзя-называть`, его все так называли даже мэр города - `Тот-кого-нельзя-оскорблять`, решил отслеживать настроение жителей. Помогите ему и напишите программу.

На вход подаётся одна строка, в которой через пробел записана последовательность целых чисел (это зарплаты горожан), например, такая: `600 100 500 1000`. Количество чисел заранее неизвестно, но их не может быть более **1000** и менее **2**-х.

На выход на экран нужно вывести **отношение самой низкой зарплаты к самой высокой, выраженное в процентах с округлением до ближайшего целого**.

Sample Input 1:

```
600 100 500 1000
```

Sample Output 1:

```
10
```

Sample Input 2:

```
100 100
```

Sample Output 2:

```
100
```

Sample Input 3:

```
100 200 300
```

Sample Output 3:

```
33
```

Sample Input 4:

```
400 300 200 100 1
```

Sample Output 4:

```
0
```

2.1.5 Шаги

Эта задача из главы "**Безумное чаепитие**" учебника "**Алгоритмы и структуры данных**".

- Ну, ладно, засиделась я с вами тут, - вздохнула Алиса, - пойду я.

- Не спеши, - резко выпалил Мартовский Заяц.

- А то что? - с угрозой парировала Алиса.

- А то никогда не вернёшься домой, - обречённо ответил Шляпник.

На Алису от неожиданности накатилась волна страха, но она и виду не подала. Она уже поняла, что с чаёвниками можно обсудить любую тему, главное не молчать и соображать.

- Но я же знаю обратную дорогу, - с осторожностью произнесла Алиса.

- В том то и дело, что чай наш заваривался на грибах и на обратном пути твой левый шаг не будет равен правому, будет меняться. Твой путь можно будет разбить на отдельные отрезки, возможно, разные по длине. Участков будет несколько - болото, тропа, пустырь, лес и т.д., а последний участок поляна со входом в кроличью нору всегда находится в тумане, поэтому нужно будет вслепую подходить ко входу в нору.

Так вот, на каждом из участков мы определим длину шага левой ногой и правой ногой в сантиметрах, а также количество шагов на участке, получится примерно вот такая таблица:

```
55 45 1000

60 62 800

58 47 1100

52 59 500
```

Если в конце пути знать насколько больше ты прошагала одной ногой по сравнению к другой, то можно будет гарантированно вернутся ко входу в нору, и ты успешно вернёшься домой!

Помогите Алисе и напишите программу д**ля вычисления разницы, полученной в конце пути**.

На вход подаются строки:

- в первой строке целое число **n** - количество участков,
- в последующих **n** строках через пробел будут записаны - длина шага левой ногой, длина шага правой ногой, количество шагов на участке.

Длины шагов подаются в сантиметрах - это целое положительное число в диапазоне **[0; 100]**, количество шагов на участке - целое число **[0; 10000]**. Количество участков в диапазоне **[0; 1000]**.

Например, для случая, указанного выше можно получить такой результат:

- Левой ногой =>55*1000 + 60*800 + 58*1100 + 52*500 = 192800 сантиметров или 1928 метров.
- Правой ногой => 45*1000 + 62*800 + 47*1100 + 59*500 = 175800 сантиметров или 1758 метров.

Таким образом, мы получаем разницу равную 170 метров. Ответ давать в метрах без знака, то есть по абсолютной величине, с округлением до ближайшего меньшего целого.

Sample Input 1:

```
4
55 45 1000
60 62 800
58 47 1100
52 59 500
```

Sample Output 1:

```
170
```

Sample Input 2:

```
1
60 65 10
```

Sample Output 2:

```
0
```

Sample Input 3:

```
1
60 65 30
```

Sample Output 3:

```
1
```

Матрицы

2.2.1 Дороги

В племени «Тумба-Юмба» есть N точек для охоты, некоторые из которых соединены тропами. Вождь племени решил провести инвентаризацию троп. Но, он не силен в математике, поэтому он просит вас сосчитать количество дорог.

Требуется написать программу, помогающую сосчитать количество дорог по заданной матрице связности точек.

Входные данные:

- в первой строке подаётся целое число N (0 ≤ N ≤ 100);
- в следующих N строках записано по N чисел, каждое из которых является единичкой или ноликом. Причем, если в позиции (i, j) квадратной матрицы стоит единичка, то i-ый и j-ый точки охоты соединены тропами, а если нолик, то не соединены.

Выходные данные:

на экран необходимо вывести число, определяющее количество троп.

Пример - если входной поток такой:

```
5
0 1 0 0 0
1 0 1 1 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
```

то правильный ответ = **3**.

Sample Input 1:

```
5
0 1 0 0 0
1 0 1 1 0
0 1 0 0 0
0 1 0 0 0
0 0 0 0 0
```

Sample Output 1:

```
3
```

Sample Input 2:

```
3
0 1 1
1 0 1
1 1 0
```

Sample Output 2:

```
3
```

2.2.2 Шамбала

Шаман племени «Тумба-Юмба» обладает особым даром – он умеет отгонять беду от посевов племени. Но для успешного камлания он должен в день начала посевов правильно сложить особенную фигуру, называемую Шамбалой, чтобы боги погоды были благосклонны к жителям племени. Как строится Шамбала посмотрите в прилагаемых к задаче примерах.

В данной задаче нужно вывести на экран в символическом виде концентрические круги символами '**X**' и **пробелами**, количество которых зависят от номера **N** текущего дня.

На вход подаётся одна строка с числом **N [1, 30]**.

На выход нужно подать несколько строк с изображением шамбалы.

Sample Input 1:

```
1
```

Sample Output 1:

```
X
```

Sample Input 2:

```
2
```

Sample Output 2:

```
XXX
X X
XXX
```

Sample Input 3:

```
3
```

Sample Output 3:

```
XXXXX
X   X
```

```
X X X
X   X
XXXXX
```

Sample Input 4:

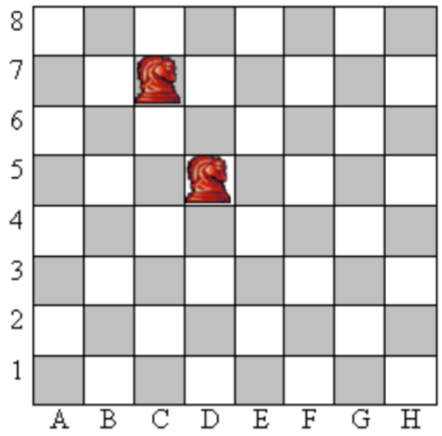
```
4
```

Sample Output 4:

```
XXXXXXX
X     X
X XXX X
X X X X
X XXX X
X     X
XXXXXXX
```

2.2.3 Ход конём

Интеллектуалы племени «Тумба-Юмба» после охоты учатся играть в шахматы. На этой неделе они осваивают ход Конём.



Напишите программу, которая будет проверять правильность хода Конём.

На вход подаётся строка – это шахматная запись одного хода - начальной позиции фигуры и конечной позиции, например, так: **D5-C7**.

На выход нужно вывести результат анализа записи хода: **YES** - если указанный ход верный, **NO** - если такой ход не по правилам, указанным для Коня.

Гарантируется, что запись хода будет состоять из 5-ти символов, в середине будет «дефис», буквы и цифры будут расположены на правильных местах и в разрешённом диапазоне для шахматной доски.

Sample Input 1:

```
D5-C7
```

Sample Output 1:

```
YES
```

Sample Input 2:

```
A1-C2
```

Sample Output 2:

```
YES
```

Sample Input 3:

```
A1-B2
```

Sample Output 3:

```
NO
```

2.2.4 Равнобедренные треугольники

Знайка иногда ведёт частные уроки по геометрии и ему нужно рисовать равнобедренные треугольники. Но это рутинная операция и Знайке совсем не хочется тратить на это своё время. Помогите ему и напишите соответствующую программу.

На вход подаётся целое число **N** в диапазоне `[1; 30]`.

На выход по результатам работы программы на экран выводится равнобедренный треугольник по такому формату:

- **N** – это высота треугольника,
- основание треугольника параллельно вертикальной стороне экрана,
- остальные параметры смотрите по примерам:

Sample Input 1:

```
1
```

Sample Output 1:

```
*
```

Sample Input 2:

```
2
```

Sample Output 2:

```
*
**
*
```

Sample Input 3:

```
3
```

Sample Output 3:

```
*
**
* *
**
*
```

Sample Input 4:

```
4
```

Sample Output 4:

```
*
**
* *
*   *
* *
**
*
```

Строки

2.3.1 Уравнение Незнайки

Уравнение для Незнайки представляет собой строку длиной 5 символов, например, такую: x+5=7.

Второй символ строки является либо знаком '**+**' (плюс) либо '**-**' (минус), четвёртый символ — знак '**=**' (равно).

Из первого, третьего и пятого символов ровно два являются цифрами из диапазона от **0** до **9**, и один — буквой **x**, обозначающей неизвестное.

Требуется написать программу, которая позволит решить данное уравнение, то есть найти величину **x**.

На вход подаётся одна строка, например: **3-x=9**.

На выход нужно подать ответ уравнения, то есть вычисленное значение **x** - для данного примера это **-6**.

Sample Input 1:

```
1+2=x
```

Sample Output 1:

```
3
```

Sample Input 2:

```
1-2=x
```

Sample Output 2:

```
-1
```

Sample Input 3:

```
x-1=0
```

Sample Output 3:

```
1
```

Sample Input 4:

```
5-x=1
```

Sample Output 4:

```
4
```

2.3.2 Шифрование Пилюлькина

Пилюлькин решил производить сладкую газировку в промышленных масштабах, но монополии запрещены в Солнечном городе. Однако он решил работать через подставные компании, а рецепт и ингредиенты каждый раз оставлял в одном из номеров солнечногорской гостиницы. Своим подельникам, Жулио и Нига, из Лос-Пога-носа он передавал лишь шифрованную запись, публикуя её в «Газете дураков».

Запись состоит из последовательности нулей и единиц без пробелов общей длиной не более **255** символов. Самая длинная непрерывная цепочка **нулей** в шифровке Пилюлькина и обозначала номер, которые должны были снять подельники в гостинице и там забрать искомые компоненты.

Например, если дана такая последовательность - **00101110000110**, то зашифрован номер **4**. Продажные полицейские солнечногорска не в состоянии разгадать такой шифр, а, возможно и не хотят, так как они продажные. Помогите детективу Биглю, не являющемуся полицейским, – напишите программу, которая по шифру Пилюлькина будет восстанавливать зашифрованный гостиничный номер.

На вход подаётся одна строка из нулей и единиц.

На выход подаётся ответ - одно целое число - зашифрованный номер.

Sample Input 1:

```
00101110000110
```

Sample Output 1:

```
4
```

Sample Input 2:

```
1110111
```

Sample Output 2:

```
1
```

Sample Input 3:

```
11000000111111010101010101
```

Sample Output 3:

```
6
```

2.3.3 Нумерология Кнопочки

Как и многие другие коротышки, малышка Кнопочка верит во всякие чудеса, обожает разные гадания и нумерологию. Кнопочка руководит солнечногорской гостиницей и считает, что некоторым особенным клиентам нужно делать значительную скидку, чтобы коротышечьи боги были благосклонны к её бизнесу.

Особенность клиентов определяется по их порядковому номеру заселения следующим образом: пусть **n** – номер заселения, тогда нужно подсчитать **сумму всех целых чисел (от 1 до n)**, на которые **n делится без остатка** и, если эта сумма окажется **нечётной**, то данный гость получает скидку - он особенный. Например: если n=3, то 1+3=4 – не особенный гость; если n=4, то 1+2+4=7 - особенный.

Кнопочка не успевает выполнять все обязанности по гостинице и ей очень нужна программа, которая автоматизирует процесс вычисления особенных клиентов.

На вход подаётся:

- в первой строке количество клиентов - целое положительное число - **n [1; 1000]**,
- в последующих **n** строках записаны через пробел номер клиента и имя клиента (имя не содержит пробела).

На выход нужно вывести на экран строки, в каждой из которых есть Номер и, через пробел, Фамилия особенного клиента. Строки нужно выдавать в порядке возрастания номеров клиентов. Кнопочка не успевает аккуратно вести журнал учёта посетителей, поэтому на вход строки будут подаваться не обязательно в порядке возрастания номеров.

Если не окажется ни одного особенного клиента, то следует вывести одну строку **empty**.

Sample Input 1:

```
12
1 Стекляшкин
2 Пилюлькин
3 Смекайло
4 Растеряйка
5 Молчун
6 Сахариныч
7 Винтик
8 Шпунтик
9 Пачкуля
10 Гусля
11 Винтик
12 Шпунтик
```

Sample Output 1:

```
1 Стекляшкин
2 Пилюлькин
4 Растеряйка
8 Шпунтик
9 Пачкуля
```

Sample Input 2:

```
4
1 Винтик
9 Пачкуля
100 Гусля
64 Шпунтик
```

Sample Output 2:

```
1 Винтик
9 Пачкуля
64 Шпунтик
100 Гусля
```

Sample Input 3:

```
1
81 Вовочка
```

Sample Output 3:

```
81 Вовочка
```

Sample Input 4:

```
2
90 Кузьмич
91 Петрович
```

Sample Output 4:

```
empty
```

2.3.4 Палиндромы

- А что будет, если мы не уложимся в точное время заваривания чая? - никак не унималась Алиса.

- Ничего хорошего не будет, - буркнул в ответ Заяц.

- Он не далёк от истины, последствия непредсказуемы, - пояснил Шляпник.

- В прошлый раз время пошло вспять, - продолжил он после паузы, занятой отхлёбыванием свежеТочноЗаваренного ароматного чая.

- Как же вы выкрутились? - удивилась Алиса.

- А также, ответил Шляпник, - мы знали заранее и подготовились, мы пользовались только числами, словами и фразами, которые и в прямом и в обратном направлении

читаются одинаково - палиндромами (данный термин имеет греческое происхождение и обозначает "движущийся обратно").

Например, число «1231» и слово «рокот» - не палиндромы, они нам не нужны. А вот, число "12321" – палиндром, слово "топот" и фраза "Аргентина манит негра!" – палиндромы. Их можно использовать при любом течении времени...

Помогите Алисе и напишите программу, которая будет проверять вводимую строку на палиндромность. При проверке на палиндромность **регистр буквы не имеет значения**, пробелы, знаки препинания и знаки пунктуации не учитываются. Учитываются только символы русского и английского алфавитов и цифры:

- АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- 0123456789

Буквы Ё нет.

Примеры палиндромов:

- На в лоб, болван.
- Я иду с мечем судия.
- А роза упала на лапу Азора.
- Аргентина манит негра!
- 12345-.,!?:;4 3 2 1

На вход подаётся одна строка - её нужно проверить на палиндромность.

На выход подайте **YES**, если строка палиндром, **NO** - иначе.

Sample Input 1:

```
На в лоб, болван.
```

Sample Output 1:

```
YES
```

Sample Input 2:

```
топор
```

Sample Output 2:

```
NO
```

Sample Input 3:

```
топот
```

Sample Output 3:

```
YES
```

2.3.5 Код пифии

Хакер Нео, не понимая своей миссии в Матрице, ищет встречи с Пифией.

Личная встреча опасна, как и любое электронное сообщение, так как Нео уже давно попал под наблюдение Матрицы, и за ним охотится Агент Смит. Чтобы передавать сообщения, Пифия размещает их в сети, но адреса документов шифрует.

Описание кода Пифии.

Вы знаете, что каждый символ имеет свой порядковый номер в таблице символов, таким образом буквы можно заменить на их номера, а слова на последовательности целых чисел. Например, рассмотрим кодировку для имени Neo. Первая буква - N в таблице символов стоит на 78-ой позиции – это и есть её номер. Но Пифия пошла дальше и стала кодировать эти номера в шестнадцатеричной системе счисления, то есть **нужно заменить 78_{10} на $4E_{16}$**, так как $4E_{16}$ => $4*16^1 + 14*16^0 = 64 + 14 = 78_{10}$ Итак, если зашифровать имя Neo кодом Пифии, то получится такая последовательность: **`4E656F`**, где каждые два последовательно записанных символа кодируют одну букву. Нео проводит дни в ожидании послания от Пифии – это будет адрес некоторого файла или сайта в сети. Однажды сообщение придёт и нужно будет оперативно его декодировать.

Помогите Избранному - напишите программу для декодирования послания от Пифии.

На вход подаётся одна зашифрованная строка.

На выход подаёте одну расшифрованную строку.

Sample Input 1:

```
68747470733A2F2F7777772E707974686F6E2E6F72672F646F776E6C6F6164
32F
```

Sample Output 1:

```
https://www.python.org/downloads/
```

Sample Input 2:

```
68747470733A2F2F6769746875622E636F6D2F
```

Sample Output 2:

```
https://github.com/
```

2.3.6 Код пифии - 2

Пифия ждёт момента, чтобы отправить послание для Нео. Агент Смит не ослабляет внимание.

Чтобы передавать сообщения, Пифия размещает их в сети, но, предварительно, адреса документов шифрует.

Описание кода Пифии.

Вы знаете, что каждый символ имеет свой порядковый номер в таблице символов, таким образом буквы можно заменить на их номера, а слова на последовательности целых чисел. Например, рассмотрим кодировку для имени Neo. Первая буква - N в таблице символов стоит на 78-ой позиции – это и есть её номер. Но Пифия пошла дальше и стала кодировать эти номера в шестнадцатеричной системе счисления, то есть **нужно заменить 78_{10} на $4E_{16}$**, так как $4E_{16}$ => $4*16^1 + 14*16^0 = 64 + 14 = 78_{10}$ Итак, если зашифровать имя Neo кодом Пифии, то получится такая последовательность: **`4E656F`**, где каждые два последовательно записанных символа кодируют одну букву.

Момент для отправки сообщения может наступить неожиданно и не будет времени вручную проводить шифровку.

Помогите Пифии - напишите программу для **кодирования** послания.

На вход подаётся одна строка.

На выход подаёте одну зашифрованную строку.

Sample Input 1:

```
https://stepik.org/61148/
```

Sample Output 1:

```
68747470733A2F2F73746570696B2E6F72672F36313134382F
```

Sample Input 2:

```
https://www.python.org/dev/peps/pep-0020/
```

Sample Output 2:

```
68747470733A2F2F7777772E707974686F6E2E6F72672F6465762F706570732
F7065702D303032302F
```

2.3.7 Точное время

Около дома под деревом стоял накрытый стол, а за столом пили чай Мартовский Заяц и Безумный Шляпник. Стол был большой, хорошо сервированный, готовый к чаепитию. Однако чаёвники сидели с пустыми чашками. Это ничуть не смутило

Алису - она выбрала место сама и, не спрашивая, уселась в удобное большое кресло.

- Приступим, - дерзко сказала Алиса.

- Мы не можем, - нервно выкрикнул Мартовский Заяц.

- Отчего же? - не унималась Алиса.

- От того, что мы не знаем, когда заварится чай... - с горечью вымолвил Безумный Шляпник.

Тягостная пауза. Алиса с недоумением смотрит на присутствующих.

- Ну, вот смотри, - нехотя продолжил Шляпник, вынимая из кармана часы.

- Сколько сейчас времени?

- Полпервого, - уверенно заявила Алиса.

- Что ты, что ты, - засуетился Мартовский Заяц, - твоя самоуверенность тебя погубит, подытожил он. Шляпник решил прояснить ситуацию: - У нас принято всё делать точно. "Сейчас не полпервого"... Мартовский кот пренебрежительно хмыкнул. - Сейчас 12.28.05, - уточнил Шляпник.

- А зачем говорить секунды? - продолжала вредничать Алиса.

- А затем, глупое создание, что наш чай нужно заваривать ровно 333 секунды и не секундой дольше.

Помогите Алисе, напишите программу, в которую на вход подаются две строки:

- начальное время в формате **XX:XX:XX** (например: 00:01:02 или 14:14:14 - всегда по два разряда на часы, минуты и секунды) и
- количество секунд **Y** для заваривания чая.

На выход следует подать конечное время в формате **XX:XX:XX**.

Чтобы найти ответ нужно к начальному времени прибавить **Y** секунд.

В этой задаче суммарное конечное время **не может** превысить **23:59:59**.

Sample Input 1:

```
00:00:00
1
```

Sample Output 1:

```
00:00:01
```

Sample Input 2:

```
00:00:58
3
```

Sample Output 2:

```
00:01:01
```

Sample Input 3:

```
09:54:28
333
```

Sample Output 3:

```
10:00:01
```

2.3.8 Точное время - 2

Эта задача аналогична предыдущей, есть только одно усложнение: в этой задаче суммарное конечное время **может** превысить **23:59:59**.

Sample Input 1:

```
00:00:00
1
```

Sample Output 1:

```
00:00:01
```

Sample Input 2:

```
23:59:59
62
```

Sample Output 2:

```
00:01:01
```

Sample Input 3:

```
09:09:09
86400
```

Sample Output 3:

```
09:09:09
```

2.3.9 Лекции Незнайки

Незнайка регулярно посещает лекции Знайки по истории, но он так неаккуратно ведёт записи, что это каждый раз становится проблемой при подготовке к экзаменам. Незнайка решил исправляться - ему нужно вести статистику своих ошибок, чтобы отслеживать улучшения. Ошибки Незнайка допускает не всегда, но когда допускает, то они состоят в том, что он в записываемом имени допускает ровно одну орфографическую ошибку, то есть заменяет **ровно одну** из букв имени какой-то другой буквой.

Пусть дан **список правильных написаний имён**, и **список имён или других слов из записей Незнайки**. Сопоставлять два списка сам Незнайка не в состоянии. Напишите программу, которая поможет вести статистику "**подозрительных слов**" из записей Незнайки.

На вход подаются строки:

- в первой строке файла находится целое число **N** – это количество правильных имён из лекций Знайки,
- следующие **N** строк содержат правильные имена,
- далее идет строка, содержащая целое число **M** – количество «**подозрительных**» имен и слов из записей Незнайки,
- следующие **M** строк – это те самые имена с ошибкой, без ошибки и иные слова.

Каждое из имен или слов – это последовательность из **K** заглавных букв английского алфавита **(1 ≤ N, M, K ≤ 30)**.

По результатам работы программы на выход на экран нужно вывести одну строку, состоящую из **N** чисел – для каждого правильного имени выводится количество «**подозрительных слов**» из записей Незнайки, то есть тех, которые **равны по длине исходному слову, но отличаются строго на одну букву**. Например, для при сравнении имени **ZEUS** и списка слов **ZEVS**, **ZEUS**, **XEUS**, **XEROX** следует вывести число 2, так как только 2 слова из списка отличаются на **ровно 1 символ**.

На выход нужно подать одну строку, в которой через пробел записаны **N** целых чисел, обозначающих количество "подозрительных слов" для соответствующих имён из лекций Знайки.

Sample Input:

```
3
ZEUS
POSEIDON
AFINA
4
ZEVS
POSEYDON
AVYNA
ZERS
```

Sample Output:

```
2 1 0
```