# Азбука Морзе

Вам дан английский текст. Закодируйте его с помощью азбуки Морзе: <https://en.wikipedia.org/wiki/Morse_code#/media/File:International_Morse_Code.svg>

Каждая буква заменяется на последовательность точек и тире. В качестве тире используйте обычный дефис: «-», а в качестве точки — точку «.». Например, буква «g» превратится в трёхсимвольную строку «--.». Между закодированными буквами ставится ровно один пробел. Например, слово «Help» превратится в «.... . .-.. .--.».

Обратите внимание, что строчные и заглавные буквы кодируются одинаково.

## Формат ввода

Весь текст записан в единственной строке. Текст состоит из английских букв и пробелов, других символов в тексте нет. В тексте не может быть двух или более пробелов подряд.

Длина текста не превышает 100 000 символов, количество слов в тексте не превышает 1000.

## Формат вывода

Выведите каждое слово исходного текста, закодированное азбукой Морзе, на отдельной строке. Количество строк в ответе должно совпадать с количеством слов в исходном тексте.

## Пример

| Ввод | Вывод |
| --- | --- |
| Help me SOS | .... . .-.. .--.  -- .  ... --- ... |

# Толковый словарь

Вася учит новые слова. Для этого он пользуется небольшим толковым словарём. Каждая запись в словаре — это слово и текстовое описание его значения.

Для проверки мама называет Васе слова из словаря — а он должен воспроизводить их описания.

Помогите Васе справиться с проверкой.

## Формат ввода

В первой строке задаётся целое число N (1 ≤ N ≤ 1000) — количество записей в толковом словаре Васи. В каждой из следующих N строк дано по одной записи: сначала идёт слово, а затем через пробел непустое описание его значения. Все слова, значение которых записано в словаре, различны.

В следующей строчке после толкового словаря записано целое число M (1 ≤ M ≤ 100) — количество слов, которое проверит мама. В следующих M строках перечислены слова для проверки, по одному на строке. Слова записаны в точности так, как они представлены в словаре.

## Формат вывода

Для каждого слова из маминого списка проверки выведите описание его значения из словаря или фразу «Нет в словаре» (без кавычек), если такого слова нет в словаре.

## Пример

| Ввод | Вывод |
| --- | --- |
| 3  Пример Поступок, поведение или явление, служащее образцом для кого-л., чего-л.  Задача Цель, к которой стремятся, которую хотят достичь.  Учёба Процесс действия по значению глаг.: учиться.  4  Пример  Отличник  Задача  Пример |  |

words\_count = int(input())

dictionary = {}

**for** i **in** range(words\_count):

word, description = input().rstrip().split(None, 1)

dictionary[word] = description

requests\_count = int(input())

**for** i **in** range(requests\_count):

word = input().strip()

**if** word **in** dictionary:

print(dictionary[word])

**else**:

print('Нет в словаре')

# 

# Проверка связи

Дана строчка русского текста, состоящая из слов и пробелов. Словом считается последовательность русских букв, слова разделены одним или большим числом пробелов.

Для каждого слова этого текста узнайте порядковый номер его вхождения в текст именно в той форме, в которой указано слово. Для первого вхождения слова выведите «1», для второго вхождения того же слова выведите «2» и так далее.

## Формат ввода

В единственной строчке записан русский текст. Длина текста не превышает 100 000 символов. Текст состоит только из русских букв и символов пробела.

## Формат вывода

Для каждого слова исходного текста выведите одно целое число — номер вхождения этого слова в текст. Числа выведите через пробел. Количество чисел должно совпадать с количеством слов в исходном тексте.

## Пример

| Ввод | Вывод |
| --- | --- |
| Раз раз раз как меня слышно Повторяю раз раз раз Повторяю | 1 1 2 1 1 1 1 3 4 5 2 |

1

words = input().split()

counter = {}

**for** word **in** words:

**if** word **not** **in** counter:

counter[word] = 1

**else**:

counter[word] += 1

print(counter[word], end=' ')

# Карта сокровищ

На пиратской карте отмечено N точек, в которых зарыты сокровища. Каждая точка задана координатами (xi, yi). Координаты указаны в километрах.

Команда Капитана Крюка хочет составить маршрут, чтобы собрать как можно больше кладов. Однако есть ограничение: для любых двух соседних точек маршрута (xi, yi) и (xj, yj) координаты xi и xj могут различаться только последней цифрой, и координаты yi и yj тоже могут различаться только последней цифрой. Например, после точки (15, 10) они могут отправиться в точку (18, 16), а вот из точки (14, 68) в точку (19, 71) пройти уже не получится — ведь 68 и 71 различаются не только последней цифрой. Из точки (5, 12) в точку (13, 14) попасть тоже нельзя, так как числа 5 и 13 отличаются в разряде десятков.

По заданным координатам определите, какое максимальное количество точек сможет добавить в свой маршрут Капитан Крюк.

## Формат ввода

В первой строке указано число N (1 ≤ N ≤ 10 000) — количество точек, отмеченных на карте сокровищ.

В следующих N строках содержатся пары координат: xi и yi — координаты i-ой точки. Координаты — целые числа не меньше нуля и не больше 1 000 000 000. Гарантируется, что совпадающих точек в списке нет.

## Формат вывода

Выведите одно число — максимальное количество точек, которое Капитан Крюк сможет посетить по маршруту, построенному по описанным правилам.

## Пример

| Ввод | Вывод |
| --- | --- |
| 9  10 18  17 15  25 21  0 21  1 16  25 29  24 24  8 26  10 20 | 3 |

points\_count = int(input())

points = {}

**for** i **in** range(points\_count):

x, y = [int(j) **for** j **in** input().split())

idx = (x // 10, y // 10)

**if** idx **not** **in** points:

points[idx] = 0

points[idx] += 1

max\_value = max(points.values())

print(max\_value)

# Дни рождения

У Васи N одноклассников. Вася не смог запомнить их дни рождения и решил составить календарь дней рождений класса. По известному списку всех дней рождения научитесь определять, у кого день рождения в заданном месяце.

## Формат ввода

В первой строчке записано целое число N (1 ≤ N ≤ 1000) — количество Васиных одноклассников. В следующих N строчках записана информация об их днях рождения. Каждая строчка состоит из трёх частей, разделённых пробелом — имени одноклассника, дня и месяца его рождения. Имя — это строка из русских букв, день — число от 1 до 31, а месяц — строка из набора «янв», «фев», «мар», «апр», «май», «июн», «июл», «авг», «сен», «окт», «ноя», «дек».

Имена всех одноклассников Васи различны.

В следующей строчке записано целое число M (1 ≤ M ≤ 100) — количество вопросов, на которое надо ответить. В следующих M строках содержатся сами вопросы. Вопрос — это название месяца в том же формате, в котором они задаются выше.

## Формат вывода

Для каждого вопроса в отдельной строчке через пробел выведите имена всех одноклассников, которые родились в указанном месяце. Имена упорядочьте в лексикографическом порядке.

Если в заданном месяце никто не родился, оставьте соответствующую строку ответа пустой.

## Пример

| Ввод | Вывод |
| --- | --- |
| 4  Ваня 20 янв  Петя 15 июн  Вася 10 янв  Коля 20 июл  3  июн  дек  янв | Петя  Ваня Вася |

birthdays = int(input())

months = {}

**for** i **in** range(birthdays):

name, \_, month = input().strip().split()

**if** month **in** months:

months[month].append(name)

**else**:

months[month] = [name]

**for** month **in** months.keys():

months[month].sort()

requests = int(input())

**for** i **in** range(requests):

month = input().strip()

**if** month **in** months:

print(' '.join(months[month]))

**else**:

print()

# Транслитерация

Дан русский текст.

Вам необходимо *транслитерировать* его, то есть заменить все русские буквы на английские по правилам транслитерации для загранпаспортов 2010 года: <http://mishka.travel/blog/index/node/id/1073-transliteraciya-v-zagranpasportah-rossiiskoi-federacii/>.

Букву «ё» транслитерируйте как «e», «й» как «и», а «ъ» и «ь» (и их заглавные версии «Ъ» и «Ь») должны исчезнуть из текста. Строчные буквы заменяются на строчные, заглавные заменяются на заглавные. Если заглавная буква превращается при транслитерации в несколько букв, то заглавной должна остаться только первая из них (например, «Ц» → «Tc»).

Все некириллические символы должны остаться на месте.

## Формат ввода

В единственной строке задан русский текст. Текст может состоять из любых символов, вам необходимо транслитерировать только русские буквы, а остальные оставить на месте. Гарантируется, что нет слов, состоящих только из букв «ъ» и «ь».

## Формат вывода

Выведите одну строку — транслитерированный текст.

## Пример

| Ввод | Вывод |
| --- | --- |
| Я помню чудное мгновенье: Передо мной явилась ты, Как мимолетное виденье, Как гений ч |  |

trans = {

'А': 'A',

'Б': 'B',

'В': 'V',

'Г': 'G',

'Д': 'D',

'Е': 'E',

'Ё': 'E',

'Ж': 'ZH',

'З': 'Z',

'И': 'I',

'Й': 'I',

'К': 'K',

'Л': 'L',

'М': 'M',

'Н': 'N',

'О': 'O',

'П': 'P',

'Р': 'R',

'С': 'S',

'Т': 'T',

'У': 'U',

'Ф': 'F',

'Х': 'KH',

'Ц': 'TC',

'Ч': 'CH',

'Ш': 'SH',

'Щ': 'SHCH',

'Ы': 'Y',

'Ъ': '',

'Ь': '',

'Э': 'E',

'Ю': 'IU',

'Я': 'IA'

}

**for** c **in** input():

u = c.upper()

**if** u **in** trans:

print(trans[u].lower() **if** c.islower() **else** trans[u].capitalize(), end='')

**else**:

print(u, end='')

# Телефонная книга

Вася решил записать телефоны всех своих друзей, чтобы потом автоматизировать поиск нужного номера.

У каждого из Васиных друзей может быть один или больше номеров телефонов. Напишите программу, которая поможет Васе находить все номера определённого друга.

## Формат ввода

В первой строке задано одно целое число N (1 ≤ N ≤ 1000) — количество номеров телефонов, информацию о которых Вася решил сохранить в телефонной книге. В следующих N строках заданы телефоны и имена их владельцев через пробел. Телефон — это несколько цифр, записанных подряд, имя же состоит только из русских букв. Записи не повторяются.

В следующей строке записано целое число M (1 ≤ M ≤ 100) — количество запросов от Васи. В следующих M строках записаны сами запросы, по одному на строке. Каждый запрос — это имя какого-то друга, чьи телефоны Вася хочет сейчас найти, записанное в точности так, как в телефонной книге.

## Формат вывода

Для каждого запроса от Васи выведите в отдельной строке все телефоны, принадлежащие человеку с этим именем. Телефоны одного человека выводите в одну строку через пробел в том порядке, в котором они были заданы во входных данных. Количество строк в ответе должно быть равно количеству запросов Васи.

Если в телефонной книге нет телефонов человека с таким именем, выведите в соответствующей строке «Нет в телефонной книге» (без кавычек).

## Пример

| Ввод | Вывод |
| --- | --- |
| 3  3129102 Ваня  79007619273 Коля  79120123456 Ваня  3  Коля  Ваня  Олег | 79007619273  3129102 79120123456  Нет в телефонной книге |

phones\_count = int(input())

phones = {}

**for** i **in** range(phones\_count):

phone, name = input().strip().split()

**if** name **in** phones:

phones[name].append(phone)

**else**:

phones[name] = [phone]

requests\_count = int(input())

**for** i **in** range(requests\_count):

name = input().strip()

**if** name **in** phones:

print(' '.join(phones[name]))

**else**:

print('Нет в телефонной книге')

# Дни рождения – 2

Вася снова составляет календарь дней рождения своих одноклассников. Теперь, зная месяц, он хочет узнавать, в какие дни этого месяца в классе будет праздник.

## Формат ввода

В первой строчке записано целое число N (1 ≤ N ≤ 1000) — количество Васиных одноклассников. В следующих N строчках записана информация об их днях рождения. Каждая строчка состоит из трёх частей, разделённых пробелом — имени одноклассника, дня и месяца его рождения. Имя — это строка из русских букв, день — число от 1 до 31, а месяц — строка из набора «янв», «фев», «мар», «апр», «май», «июн», «июл», «авг», «сен», «окт», «ноя», «дек».

Имена всех одноклассников Васи различны.

В следующей строчке записано целое число M (1 ≤ M ≤ 100) — количество вопросов, на которое надо ответить. В следующих M строках содержатся сами вопросы. Вопрос — это название месяца в том же формате, в котором они задаются выше.

## Формат вывода

Для каждого вопроса в отдельной строчке через пробел выведите имена всех одноклассников и дни их рождения, которые родились в заданном месяце. Информацию об именинниках упорядочьте по дню рождения. Если два человека родились в один день одного месяца, выведите их в лексикографическом порядке имён.

Если в заданном месяце никто не родился, оставьте соответствующую строку ответа пустой.

## Пример

| Ввод | Вывод |
| --- | --- |
| 4  Ваня 20 янв  Петя 15 июн  Вася 10 янв  Коля 20 июл  3  июн  дек  янв | Петя 15  Вася 10 Ваня 20 |

birthdays = int(input())

months = {}

**for** i **in** range(birthdays):

name, day, month = input().strip().split()

day = int(day)

**if** month **in** months:

months[month].append((day, name))

**else**:

months[month] = [(day, name)]

**for** month **in** months.keys():

months[month].sort()

requests = int(input())

**for** i **in** range(requests):

month = input().strip()

**if** month **in** months:

print(' '.join(pair[1] + ' ' + str(pair[0]) **for** pair **in** months[month]))

**else**:

print()

# Права доступа

В операционных системах семейства Unix пути к файлам имеют следующий вид: /folder1/folder2/another\_folder/filename.ext. Они начинаются с символа «/» (слэш), затем идут имена вложенных папок, разделённые этим же символом. Например, в нашем примере папка another\_folder вложена в папку folder2, которая, в свою очередь, вложена в папку folder1. В конце, после последнего слэша, идёт имя файла и расширение («filename.ext»), разделённые точкой.

Вам поручено написать систему распределения доступа к файлам. Она должна не пускать пользователя к тем файлам, прочитать которые у него не хватает прав.

Права пользователю можно выдавать как на отдельный файл, так и на целую папку. Например, можно разрешить читать только файл /folder1/folder2/another\_folder/filename.ext, а можно разрешить читать всю папку /folder1/folder2/ и тогда пользователь сможет автоматически прочитать любой файл в этой папке, в любой её подпапке и так далее.

Вашей программе выдан список разрешений и запросов. Для каждого запроса скажите, имеет ли право пользователь читать этот файл. Пользователь имеет право читать файл, если ему явно разрешили читать именно этот файл, или если ему разрешили читать любую из папок, в которых находится этот файл. В примере выше пользователь может читать файл filename.ext, если ему разрешили читать его или если ему разрешили читать одну из папок /folder1, /folder1/folder2, /folder1/folder2/another\_folder.

## Формат ввода

В первой строчке дано целое число N (1 ≤ N ≤ 1000) — количество выданных разрешений. В следующих N строчках записаны пути, которые пользователю разрешено читать. Путь может содержать только латинские буквы (заглавные и строчные), символ подчёркивания (\_), точку и слэш. Гарантируется, что подпапка и файл в одной папке не могут иметь одинаковое название.

В следующей строчке дано целое число M (1 ≤ M ≤ 1000) — количество запросов. В следующих M строчках записаны пути, для которых запрашивается доступ пользователя.

## Формат вывода

Для каждого запроса выведите в отдельной строке «YES» или «NO» (без кавычек). Выведите «YES», если пользователь сможет прочитать этот файл (то есть ему был выдан доступ к файлу или к одной из его родительских папок), и «NO», если не сможет.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| 1  /Temp/Data  1  /Secret/Temp/Data | NO |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| 1  /User/Documents/Secret/Data  1  /Secret/Data | NO |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| 3  /home/alex  /home/ivan/Documents  /var/www  3  /home/alex/HarryPotter8.doc  /home/ivan\_urgant/game.exe  /www/index.html | YES  NO  NO |

n = int(input())

*# Здесь можно обойтись множеством (set) вместо словаря*

paths = {}

**for** i **in** range(n):

paths[input().strip()] = True

requests = int(input())

**for** i **in** range(requests):

path = input().strip().split('/')

has\_access = False

acc = ''

**for** folder **in** path:

**if** folder == '':

**continue**

acc += '/' + folder

**if** acc **in** paths:

has\_access = True

print('YES' **if** has\_access **else** 'NO')

# Характеристики двоичных чисел

В строке через пробел записано некоторое количество целых десятичных чисел.

Напишите программу, которая составит список словарей, в которых каждое число будет характеризоваться количеством разрядов, нулей и единиц в двоичном представлении. Формат вывода смотрите в примерах.

## Формат ввода

Строка из целых чисел, разделённых пробелами.

## Формат вывода

Вывести список словарей, в котором для каждого из чисел записаны его характеристики по ключам:

* *digits* – количество разрядов в двоичном представлении числа;
* *units* – количество единиц в двоичном представлении числа;
* *zeros* – количество нулей там же.

## Пример

| Ввод | Вывод |
| --- | --- |
| 5 8 12 | [{'digits': 3, 'units': 2, 'zeros': 1}, {'digits': 4, 'units': 1, 'zeros': 3}, {'digits': 4, 'units': 2, 'zeros': 2}] |

numbers = [int(x) **for** x **in** input().split()]

result = []

**for** number **in** numbers:

temp = {'digits': 0, 'units': 0, 'zeros': 0}

**while** number:

d = number % 2

**if** d:

temp['units'] += 1

**else**:

temp['zeros'] += 1

temp['digits'] += 1

number //= 2

result.append(temp)

print(result)

# Репосты

Стас — заядлый пользователь социальной сети «ВДрузьях». Как только он видит интересный пост в каком-нибудь паблике, он сразу репостит его к себе на стену. Иногда его друзья репостят к себе эти репосты из пабликов. Один и тот же пост может быть опубликован много-много раз!

Когда какой-нибудь пользователь социальной сети видит один из этих репостов, то некоторая величина (назовём её *популярность*) конкретно этого репоста увеличивается на единицу. На единицу также увеличивается популярность у того репоста, откуда был сделан этот репост, а также у того репоста, откуда был сделан этот предыдущий репост, и так далее по цепочке, до оригинального поста в паблике. Чем длиннее была цепочка репостов, тем у большего количества публикаций увеличится популярность (но у каждой публикации она увеличится ровно на единицу).

Пользователей бесят повторяющиеся посты в ленте, поэтому одному пользователю не показываются даже разные репосты одного и того же оригинального поста.

Известно, сколько раз каждый из репостов показывался пользователям. Посчитайте популярность каждого из них.

## Формат ввода

В первой строке содержится целое число N (1 ≤ N ≤ 100) — количество записей в истории сайта. В следующих N строках содержатся сами записи.

Первая запись имеет вид «*Название\_паблика* опубликовал пост, количество просмотров: *X*». Остальные записи имеют вид «*Имя\_человека* отрепостил пост у *Имя\_человека или Название\_паблика*, количество просмотров: Y».

Записи в истории даны в хронологическом порядке. Название паблика и имена людей состоят только из русских букв. Имена всех репостивших пост людей различны и отличаются от названия паблика.

## Формат вывода

Выведите N строк: для оригинального поста из паблика и каждого репоста выведите одно число — его популярность. Популярности выведите в хронологическом порядке репостов, то есть в том же порядке, в котором они даны во входном файле.

## Пример

| Ввод | Вывод |
| --- | --- |
| 5  ЯндексЛицей опубликовал пост, количество просмотров: 15  Иван отрепостил пост у ЯндексЛицей, количество просмотров: 40  Василий отрепостил пост у ЯндексЛицей, количество просмотров: 13  ОлегПетрович отрепостил пост у ЯндексЛицей, количество просмотров: 39  ПапаЖени отрепостил пост у ОлегПетрович, количество просмотров: 21 | 128  40  13  60  21 |

n = int(input())

popularity = {}

first\_line = input().split()

public\_name = first\_line[0]

popularity[public\_name] = int(first\_line[-1])

names = [public\_name]

lines = []

**for** i **in** range(n - 1):

line = input().split()

lines.append(line)

names.append(line[0])

**for** line **in** reversed(lines):

name = line[0]

reposted = line[4][:-1]

views = int(line[-1])

**if** name **in** popularity:

popularity[name] += views

**else**:

popularity[name] = views

**if** reposted **in** popularity:

popularity[reposted] += popularity[name]

**else**:

popularity[reposted] = popularity[name]

**for** name **in** names:

print(popularity[name])

# Частотный анализ – 1

Дан текст на русском языке. Посчитайте, сколько раз встречается в тексте каждое слово.

Слово — это последовательность русских букв. Кроме слов, в тексте могут присутствовать пробелы и знаки препинания («,», «.», «!», «?», «:» и «;»). Других символов в тексте нет.

Слова необходимо сравнивать с точностью до регистра. То есть «Привет» и «привет» — это одно и то же слово, а вот «рыба», «рыбы» и «рыбу» — три разных.

## Формат ввода

В первой строчке задано одно целое число N (1 ≤ N ≤ 1000) — количество строк в тексте. В следующих N строках приведён сам русский текст. Текст состоит из русских букв, пробелов и знаков препинания: «,», «.», «!», «?», «:» и «;». Длина текста не превышает 100 000 символов.

## Формат вывода

Выведите слова исходного текста, упорядоченные по убыванию количества их попаданий в текст. В случае, если два слова встречаются в тексте одинаковое количество раз, выведите их в лексикографическом порядке.

Каждое слово нужно вывести в отдельной строке. Первая буква слова должна быть заглавной (даже если она не была заглавной в исходном тексте), остальные — строчными.

## Пример

| Ввод | Вывод |
| --- | --- |
| 2  Ехал Грека через реку. Видит Грека в реке рак.  Сунул Грека руку в реку, рак за руку Греку цап. | Грека  В  Рак  Реку  Руку  Видит  Греку  Ехал  За  Реке  Сунул  Цап  Через |

words = {}

**for** i **in** range(int(input())):

text = input().replace('.', '').replace(',', '').replace('!', '').replace('?', '').replace(

':', '').replace(';', '').split()

**for** word **in** text:

**if** word.lower().capitalize() **in** words:

words[word.lower().capitalize()] += 1

**else**:

words[word.lower().capitalize()] = 1

full\_keys = []

**for** count **in** list(words.values()):

**if** count **not** **in** full\_keys:

full\_keys.append(count)

full\_keys.sort(reverse=True)

list\_of\_keys = []

**for** key **in** full\_keys:

temp = [word **for** word **in** list(words.keys()) **if** words[word] == key]

temp.sort()

list\_of\_keys.append(temp)

**for** key **in** list\_of\_keys:

**for** item **in** key:

print(item)

# Радиоактивная порода

При добыче полезных ископаемых геологи обнаружили пласт с радиоактивными элементами. Отгружать излучающую породу нельзя, придется ждать, пока радиоактивный фон не снизится до приемлемого уровня.

Необходимо написать программу, которая определит, через какое время суммарный фон, создаваемый радиоактивными элементами пласта, снизится до допустимого уровня.

## Формат ввода

Строка, в которой для каждого радиоактивного элемента через пробел записан его период полураспада (целое число дней).

Строка — распределение элементов в пласте породы.

Строка — активность каждого элемента из предыдущей строки.

Число — допустимый радиационный фон.

## Формат вывода

Целое число — через какое количество дней суммарный радиационный фон снизится до допустимого уровня.

Список распределения конечной активности элементов в том же порядке.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| A 10 B 15  A B  0.8 0.6  0.9 | 15  0.4 0.3 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| W 20 L 15 Hp 25  L W Hp W L  0.2 0.14 0.5 0.3 0.2  0.5 | 40  0.05 0.035 0.25 0.075 0.05 |

## Примечания

Период полураспада — это время, через которое активность элемента уменьшается в 2 раза. Считать, что изменение происходит не постепенно, а мгновенно в тот день, когда проходит период полураспада.

У одного и того же элемента в разных местах пласта может быть разная активность.

День, на который приведены значения, считается нулевым.

elements\_data = input().split()

elements = {elements\_data[i]: int(elements\_data[i + 1])

**for** i **in** range(0, len(elements\_data), 2)}

composition = input().split()

activity = [float(layer\_activity) **for** layer\_activity **in** input().split()]

normal\_activity = float(input())

day = 0

current\_activity = sum(activity)

**while** current\_activity > normal\_activity:

day += 1

**for** layer\_num **in** range(len(composition)):

**if** day % elements[composition[layer\_num]] == 0:

activity[layer\_num] /= 2

current\_activity = sum(activity)

print(day)

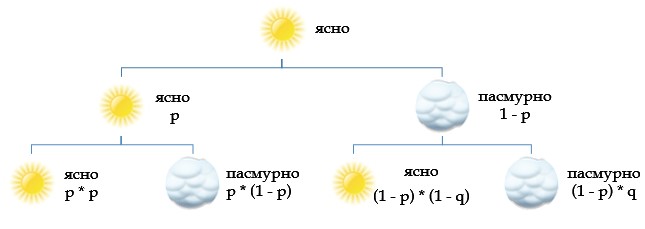
print(' '.join(str(layer\_activity) **for** layer\_activity **in** activity))

# 

# Предсказание погоды с памятью

Самый простой случай в статистике — это когда нужно определить вероятность появления одного из равновероятных событий. Например, выпадения орла или решки при броске монеты. Если бросать монету несколько раз, то вероятность не изменится, монета ничего не знает о результатах наших предыдущих экспериментов.  
Но так бывает не всегда. Замечено, что после пасмурного дня скорее всего будет тоже пасмурный, а после ясного — ясный. Допустим, что мы много лет наблюдали за погодой и накопили большой объем статистики и определили вероятность ясного дня после ясного как **p**, соответственно, пасмурного после ясного как **1 — p**; пасмурного после пасмурного как **q**, тогда ясного после пасмурного **1 — q**.

Чтобы посчитать вероятность ясного дня, нам нужно просуммировать вероятности возникновения всех ясных дней при любой погоде в предыдущие дни. Но в этой задаче мы хотим узнать максимальную из этих вероятностей.

Пусть сегодня ясно. Тогда завтра ясно будет с вероятностью **p**, а пасмурно — **1** **— p**. А послезавтра?  
  
Остается только определить максимальную вероятность интересующей нас погоды из полученных. Так можно попытаться предсказать погоду на любой день после сегодняшнего. Напишите для этого программу.

## Формат ввода

Строка — какая погода сегодня.  
Вещественное число — вероятность ясного дня после ясного.  
Вещественное число — вероятность пасмурного дня после пасмурного.  
Целое число — через сколько дней нам нужен прогноз.

## Формат вывода

Вывести прогноз на интересующий нас день — **ясно**или **пасмурно**, а также полученную вероятность. Если вероятность одинакова, вывести — **равновероятно**.

### Пример 1

| Ввод | Вывод |
| --- | --- |
| ясно  0.9  0.75  2 | ясно  0.81 |

### Пример 2

| Ввод | Вывод |
| --- | --- |
| пасмурно  0.6  0.4  3 | ясно  0.216 |

### Пример 3

| Ввод | Вывод |
| --- | --- |
| пасмурно  0.5  0.5  3 | равновероятно  0.125 |

today = input()

p = float(input())

q = float(input())

n = int(input())

**if** today == 'ясно':

predict = {'ясно': p, 'пасмурно': 1 - p}

**else**:

predict = {'ясно': 1 - q, 'пасмурно': q}

**for** \_ **in** range(1, n):

predict['ясно'] = max(predict['ясно'] \* p, predict['пасмурно'] \* (1 - q))

predict['пасмурно'] = max(predict['пасмурно'] \* q, predict['ясно'] \* (1 - p))

**if** predict['ясно'] > predict['пасмурно']:

print('ясно', predict['ясно'], sep='\n')

**elif** predict['ясно'] < predict['пасмурно']:

print('пасмурно', predict['пасмурно'], sep='\n')

**else**:

print('равновероятно', predict['пасмурно'], sep='\n')