A. Свой-Чужой

Датой освоения космоса является 4 октября 1957 года, – это день, когда Советский Союз в рамках своей космической программы первым запустил космический аппарат – «Спутник-1». В этот день шарообразный спутник вышел на орбиту, передав обратно сигнал об успешном старте. В след за СССР 1 февраля 1958 года США запустила свой первый спутник «Эксплорер-1». Таким образом началась космическая гонка. Для определения своих спутников была разработана система распознавания сигналов, которая работает по схеме «Свой-Чужой». Один из спутников отправляет запрос другому спутнику в формате двух целых чисел, а второй спутник отвечает первому так же двумя целыми числами. Первые два числа первого спутника представляют собой количество цифр и сумму цифр тех двух чисел, которыми должен ответить второй спутник. При этом в качестве ответа должны получиться различные числа, представляющие наибольшее и наименьшее возможные значения, которые могут быть сформированы по описанному выше методу.

Вам предстоит написать программу, формирующую ответ для второго спутника по известным числам, полученным от первого спутника. И определить свой это спутник или чужой.

Формат ввода

Вводятся два натуральных числа K и S, представляющих количество и сумму цифр соответственно ($0 < K \leq 5$). При этом гарантируется, что возможно составить хотя бы одно K-значное число, сумма цифр которого равна S.

Формат вывода

Выведите два числа – ответ второго спутника в строчку через пробел. На второй строке слово «Свой», если максимальное и минимальное число различны, в противном случае «Чужой».

Пример 1

3 1	100 100 Чужой

Пример 2

3 2	200 101 Свой

Примечания

При этом следует помнить, что все числа не имеют лидирующих нулей.

Решение1:

```
from itertools import product

K, S = map(int, input().split())

data = product("0123456789", repeat=K)
data = filter(lambda x: sum(list(map(int, x))) == S and len(str(int("".join(x)))) == K, data)
data = list(map(lambda x: int("".join(x)), data))

max_ = max(data)
min_ = min(data)
print(max_, min_)
if max_ == min_:
    print("Чужой")
else:
    print("Свой")
```

Решение2:

```python
a, b = map(int, input().split())
m1 = 10 ** 9
m2 = -1
for k in range(10 ** (a - 1), 10 ** a):
    if sum(int(c) for c in str(k)) == b:
        m1 = min(m1, k)
        m2 = max(m2, k)
print(m2, m1)
if m1 == m2:
    print('Чужой')
else:
    print('Свой')
```

B. Отбор в космонавты

После полета Юрия Гагарина в 1961 практически каждый мальчик СССР хотел стать космонавтом. Прошло уже более полувека, но профессия космонавт все так же престижна. К сожалению, не каждый желающий может пройти отбор, существуют высокие требования к уровню подготовки будущих космонавтов, а также ограничения по антропометрическим показателям. Например, рост космонавта не может быть больше 190 см и меньше 150 см.

Напишите программу, которая считывает рост претендентов в отряд космонавтов до тех пор, пока не будет введен «!». А затем выводит на первой строчке количество подходящих кандидатур, а на второй строке – минимальный и максимальный рост участников, отобранных в новый отряд космонавтов.

Гарантируется, что в отряд отберутся как минимум два летчика-космонавта.

Формат ввода

Несколько строк с ростом космонавтов и последняя строка «!».

Формат вывода

Две строки: количество кандидатур на первой, и минимальный и максимальный рост через пробел – на второй.

Пример

192 189 145 162 172 !	3 162 189

Решение1:

```
sp = []
n = input()
while n != "!":
    if 150 <= int(n) <= 190:
        sp.append(int(n))
    n = input()
print(len(sp))
print(min(sp), max(sp))
```

C. Мастер на все руки

Семейный дом Титовых («Ласточкино гнездо») Степан Павлович рубил сам. Репродукции картин – его. На самодельном комоде часы, вмонтированные в кремлевскую башню, – тоже сам мастерил. А вот изделие Германа – узорчатая подставка для шкатулки. Все Титовы – искусные умельцы. Пожалуй, нет такого музыкального инструмента, на котором бы Степан Павлович не играл. Тонкие, чуткие у него пальцы, владеют они и кистью, и скрипкой, а доведись – умело держат топор, рубанок, баранку автомашины, рычаги трактора… Выйдешь из дома – попадешь в сад. Тут и смородина с малиной, и вишня, и яблони. И все разных сортов, иные и без названия пока, потому что выведены самим Степаном Павловичем. И вот что примечательно: подобные сорта ягод и фруктов встретишь по всему Полковниково. «Заразил» Степан Павлович односельчан разведением садов, любого саженцами оделял, наказывая: «И соседа присылайте. Всем хватит». «Ласточкино гнездо» стоит на улице Титова – длинная и ровная, она тянется вдоль всего села. Похожа на ось координат. И не счесть, сколько титовских яблонек растет в соседних оградах! Разве что записи помогут разобраться.

Если представить, что улица Титова действительно ось абсцисс (X), то каждый дом этой улицы можно обозначить координатой – целым положительным числом. И у каждого дома – свое буквенное обозначение, отвечающее за конкретный сорт яблони, которую подарил хозяевам Титов-старший. Эти данные есть в записях, которые предоставлены исследователям-селекционерам для анализа. Задача исследователей понять, насколько дома с одинаковыми сортами яблонь далеко друг от друга находятся.

Формат ввода

На вход программе подаются координаты домов (по возрастанию), разделенные символом пробела.

Затем на следующей строке подаются сорта яблонь, так же разделенные символом пробела.

Формат вывода

Программа должна вывести одно число — максимальное расстояние между домами с одинаковыми сортами яблонь.

Если повторяющихся сортов нет, то следует вывести 0.

Пример 1

2 5 10 11 16 17 18 20 r w g w g r w b	15

Пример 2

1 3 5 6 10 21 22 r g b w w r g	20

Примечания

Максимальное количество домов и сортов яблонь в тестах равно 100000 и 100.

Решение1:

```
address = list(map(int, input().split()))
color = input().split()

sl = {}
for i in range(len(color)):
    if color[i] not in sl:
        sl[color[i]] = [address[i]]
    else:
        sl[color[i]] += [address[i]]

max_hause = 0
for value in sl.values():
```

```
    if len(value) > 1:
        if max(value) - min(value) > max_hause:
            max_hause = max(value) - min(value)
print(max_hause)
```

Решение2:

```
a = list(map(int, input().split()))
b = list(map(str, input().split()))
s = 0
for i in range(len(b) - 1):
    for j in range(i + 1, len(b)):
        if b[j] == b[i] and (a[j] - a[i]) > s:
            s = a[j] - a[i]
print(s)
```

D. Суеверия космического наставника

Если «земного» отца и наставника космонавта Германа Титова знали все советские граждане, то имя «космического» отца и наставника, Сергея Королёва, было засекречено до самой смерти конструктора. И только спустя много лет людям открылись некоторые факты из жизни гениального инженера.

Сергей Павлович Королёв был весьма суеверным человеком. В саду около его дома висела подкова на счастье, а после нескольких неудачных стартов, случившихся в понедельник, он решил больше не переносить полеты на этот день недели. К тому же он не любил, когда на площадке во время запусков были женщины, а в кармане держал две монеты — их он перебирал, когда сильно нервничал.

Представим, если бы у Сергея Королёва были любимые числа (например, 3 и 7), от которых зависели бы даты полетов. Напишите программу, которая подсчитает удачные даты стартов, исходя из одной счастливой даты (последнего успешного полета), которая вводится в первой строке.

Известно, что это не может быть тот же день недели, что и в предыдущий раз.

Известно, что запуск никогда не планируется на следующий день.

Известно, что сумма всех цифр в номерах дня и месяца должна делиться на 3 или 7.

Формат ввода

Дата в формате DD.MM.YYYY – день последнего удачного полета.

Целое число – количество новых полетов, которые нужно рассчитать.

Формат вывода

Вывести необходимое количество подходящих дат, каждую с новой строки, в виде, показанном в примерах.

Пример 1

23.02.2021 3	25 Feb 2021 28 Feb 2021 03 Mar 2021

Пример 2

29.04.2021 5	01 May 2021 04 May 2021 07 May 2021 09 May 2021 11 May 2021

Решение1:

```
from datetime import *

d, m, g = map(int, input().split("."))
dat = date(g, m, d)
n = int(input())
dat += timedelta(days=1)

while n != 0:
    dat += timedelta(days=1)
    f = sum(map(int, (str(dat.day) + str(dat.month))))
    if f % 3 == 0 or f % 7 == 0:
        print(dat.strftime("%d %b %Y"))
        n -= 1
        dat += timedelta(days=1)
```

E. База данных полетов

На 1 января 2023 года в космосе побывало 597 человек. Из них 72 космонавта СССР, 59 космонавтов России. Как и почему именно Юрия Гагарина и Германа Титова выбрали на роль советских космических первопроходцев, точно неизвестно. Однако командирам лётных полков тех лет приходили распоряжения: «Направить в такую-то часть молодого лётчика, морально стойкого, политически грамотного, с определёнными физическими и медицинскими показателями». Те, кто подходил под все критерии, в итоге направлялись в секретный полк – их начинали готовить к космическим полетам.

Вам доступна часть базы данных полетов в космос. Таблица из базы представлена в специальном виде, через запятую без пробелов указаны:

- Дата полета
- Корабль
- Экипаж
- Страна

Формат ввода

На вход подаются данные, представляющие собой таблицу. Ее значения разделены запятыми.

В первой строке вводится заголовок таблицы flight_date, ship, crew, country (дата полета, корабль, экипаж, страна).

Затем вводится заранее неизвестное количество строк со значениями этих колонок.

Формат вывода

На основе полученных данных создайте и выведите словарь с ключами годов полетов, словарей с ключами day_month (день месяц), а значением словарь с ключами ship (корабль), crew (экипаж) значения отсортированы по алфавиту, country (страна).

Пример

flight_date, ship, crew, country 12 апреля 1961, Восток, Гагарин А.Ю., СССР 5 мая 1961, Фридэм-7, Шепард А., США 11 августа 1962, Восток-3, Николаев А.Г., СССР 12 октября 1964, Восход-1, Комар В.М.; Феоктистов К.П.; Егоров Б.Б., СССР 18 марта 1965, Восход-2, Беляев П.И.; Леонов А.А., СССР 31 января 1971, Апполон-14, Шепард А.; Руса С.; Митчелл Э., США	{'1961': {'12 апреля': {'ship': 'Восток', 'crew': ['Гагарин А.Ю.'], 'country': 'СССР'}, '5 мая': {'ship': 'Фридэм-7', 'crew': ['Шепард А.'], 'country': 'США'}}, '1962': {'11 августа': {'ship': 'Восток-3', 'crew': ['Николаев А.Г.'], 'country': 'СССР'}}, '1964': {'12 октября': {'ship': 'Восход-1', 'crew': ['Егоров Б.Б.', 'Комар В.М.', 'Феоктистов К.П.'], 'country': 'СССР'}}, '1965': {'18 марта': {'ship': 'Восход-2', 'crew': ['Беляев П.И.', 'Леонов А.А.'], 'country': 'СССР'}}, '1971': {'31 января': {'ship': 'Апполон-14', 'crew': ['Митчелл Э.', 'Руса С.', 'Шепард А.'], 'country': 'США'}}}

Примечания

Если экипаж состоит из нескольких человек, то их разделяют точкой запятой

Решение1:

```
import sys

data = list(map(str.strip, sys.stdin))[1:]

sl = {}
sl_key = {}
for i in data:
```

```
    k = i.split(", ")
    if k[0].split()[-1] not in sl:
        sl_key[" ".join(k[0].split()[:-1])] = {"ship": k[1],  "crew": sorted(k[2].split("; ")),  "country": k[-1]}
        sl[k[0].split()[-1]] = {" ".join(k[0].split()[:-1]): sl_key[" ".join(k[0].split()[:-1])]}
    else:
        sl_key[" ".join(k[0].split()[:-1])] = {"ship": k[1], "crew": sorted(k[2].split("; ")), "country": k[-1]}
        sl[k[0].split()[-1]][" ".join(k[0].split()[:-1])] = sl_key[" ".join(k[0].split()[:-1])]

print(sl)
```

Решение2:

```
from sys import stdin

lines = []
for line in stdin:
    lines.append(line)

d = dict()
for i in range(1, len(lines)):
    values = lines[i].split(", ")
    day, month, year = values[0].split()
    if year not in d:
        d[year] = dict()
    d[year][day + " " + month] = {"ship": values[1], "crew": sorted(values[2].split("; ")),
                                  "country": values[3].replace("\n", "")}
print(d)
```

F. Сортировка почты

На борту российского сегмента Международной космической станции существует почтовое отделение с полагающимся штемпелем, которым гасят, как положено, почтовую корреспонденцию.Возникла космическая почта более сорока лет назад. Поток писем и посылок космонавтам был столь большим, что в 1965 году на основании директивы Главного штаба Военно-воздушных сил на почте Зеленого (будущего Звездного) городка создали специальный отдел, работавший с почтовыми отправлениями, на которых стояла только фамилия космонавта.Через какое-то время отдел вырос в информационную группу в структуре Центра подготовки космонавтов, а когда сведения о космонавтах стали благодаря гласности более доступными, ее переименовали в «Почту летчиков-космонавтов».

Космонавтам письма слали со всех уголков мира. Всю корреспонденцию, которая приходила на имя второго космонавта СССР, Германа Степановича Титова, читал лично его отец, Степан Павлович Титов. В алтайской сельской школе он отработал 30 лет и знал иностранные языки. Но особенно увлекался эсперанто. Письма для других космонавтов сортировались и направлялись адресатам.

В качестве одного из алгоритмов сортировки писем предлагалось реализовать упорядочивание фамилий по количеству гласных букв не учитывая регистр.Напишите алгоритм, который позволил бы почтовой службе отсортировать фамилии в списке по количеству гласных букв без учета регистра, в случае одинакового количества по убыванию длины не считая одинаковые буквы.

Формат ввода

В первой строке задается число n – количество писем.

Во второй строке через # окруженный двух сторон пробелами a1, a2, ..., a3 – элементы списка.

Формат вывода

Выведите через пробел отсортированный список фамилий.

Пример

5 Гагарин # Титов # Жуков # Павлов # Шевченко	Жуков Павлов Титов Шевченко Гагарин

Решение1:

```
def kol_bukvi(n):
    st = 'ауоыэяюёие'
    return sum(1 for i in n if i.lower() in st)


n = int(input())
sp = input().split(" # ")
sp.sort(key=lambda x: (kol_bukvi(x), -len(set(x.lower()))))
print(*sp)
```

G. Фильтр против худого

Степан Павлович Титов был настолько самобытным, грамотным, увлеченным человеком, что в Алтайском крае его именем названа премия, которая вручается лучшим сельским педагогам-новаторам, кто ведет активную просветительскую работу в области литературы и искусства, семейного воспитания, краеведения, пропагандирует передовой педагогический опыт. Летом 1969 года на встрече в редакции «Комсомольской правды» с учителями, едущими работать в село, Степан Павлович говорил так: «Обилие теле- и радиоинформации – несомненное благо наше, однако и тут надо помочь ребенку разобраться, отделить нужное от наносного, серьезное от пустой забавы, научить смотреть и слушать. Это сделает учитель, так как его работа связана с таким периодом жизни человека, когда управление ростом – самое серьезное дело… Учитель стоит как бы у колыбели формирования понятий ребенка. Он же кажется мне своеобразным фильтром против худого, что просачивается в коллектив детей… Не ошибусь, если скажу, что от учителя во многом зависят моральное здоровье его питомцев, история нашего государства». Напишите программу, которая отделяет нужную и полезную информацию от вредной и излишней. Для оценки «нужно/не нужно» вам пригодится проверочная строка.

Формат ввода

Вводится проверочная строка (2 символа), затем вводятся строки, пока не будет введена строка МЕД.

Формат вывода

Из каждой строки нужно выбрать часть, заключенную между предпоследним и последним вхождениями проверочной строки в строку (проверочную не включая). Полученные строки вывести без повторений, каждую с новой строки, в алфавитном порядке. Если строк с как минимум двумя вхождениями проверочной строки не нашлось, вывести -2.

Пример 1

да просила да не до просилась да вскоре все у нее получилось дарила мать сыну дорогую кофту да на сына она не налезла просьба сына к матери МЕД	не до просилась рила мать сыну дорогую кофту

Пример 2

DF sdgshdgSDshdgsdDF dfdfjhfjsdfhs Dfdfdlkfdk xfdfjhdfj DFkjidjdfdj qwertyuioprtyuioplkjhgfvbnmdsdfgsDFdfdfhdjfd fdgfhjdf dhdfhdf dfhfgyfgdjks МЕД	dfdfjhfjsdfhs Dfdfdlkfdk xfdfjhdfj

Решение1:

```
proverka = input()[::-1]
word = input()[::-1]
flag = True
mn = set()
while word[::-1] != "МЕД":
    n = 0
    k = 0
    for i in range(0, len(word)):
        if word[i:i + 2] == proverka and n == 0:
            n = i + 2
        elif word[i:i + 2] == proverka and k == 0:
```

```
            k = i
    if n != 0 and k != 0:
        mn.add(word[n:k][::-1])
        flag = False
    word = input()[::-1]
if flag:
    print(-2)
else:
    print(*sorted(list(mn)), sep="\n")
```

Решение2:

```
proverka = input()
rez, s = [], ''
while s != 'МЕД':
    word = [i for i in s.split(proverka)]
    if len(word) > 2:
        rez.append(word[-2])
    s = input()
if len(rez) > 0:
    answer = sorted(set(rez))
    for i in range(len(answer)):
        print(answer[i])
else:
    print(-2)
```

H. Проверка корректности

Чтобы стать наставником космонавта, нужно иметь не только природой данные таланты, но и особую выдержку, точность во всем. Сколько учеников Степана Павловича стали уважаемыми людьми, сколько стали педагогами – пошли по стопам своего учителя. И сейчас они вспоминают с теплом свои школьные годы. И держат друг с другом связь. Вот только бывает, что в телефонной книге чей-то номер записан неточно. Помогите выявить и удалить неверно записанные номера из базы данных.

Номер телефона считается корректным если он имеет следующий формат:
--**** или 7-***-***-****, где вместо * – цифры от 0 до 9.

Формат ввода

На вход программе подается строка текста.

Формат вывода

Программа должна вывести «YES» если строка является корректным телефонным номером и «NO» в противном случае.

Пример 1

3A1-4M7-582B	NO

Пример 2

301-447-5820	YES

Примечания

Телефонный номер должен содержать только цифры и символ -, а количество цифр в каждой группе должны быть правильным.

Решение1:

```
line = input().split("-")
if len(line) == 3:
    ok = line[0].isdigit() and line[1].isdigit() and line[2].isdigit()
    if len(line[0]) == len(line[1]) == 3 and len(line[2]) == 4 and ok:
        print("YES")
    else:
        print("NO")
elif len(line) == 4:
    ok = line[1].isdigit() and line[2].isdigit() and line[3].isdigit()
    if line[0] == '7' and len(line[1]) == len(line[2]) == 3 and len(line[3]) == 4 and ok:
        print("YES")
    else:
        print("NO")
else:
    print("NO")
```

Решение2:

```
import re

r = re.fullmatch(r"(7-)?\d{3}-\d{3}-\d{4}", input())
if r is None:
    print("NO")
else:
    print("YES")
```

Решение3:

```
s = input().split('-')
a = []
for i in s:
    a.append(len(i))
if a == [3, 3, 4] and ''.join(s).isdigit():
    print('YES')
else:
    if a == [1, 3, 3, 4] and ''.join(s).isdigit() and s[0] == '7':
        print('YES')
    else:
        print('NO')
```

Решение4:

```
tel = input()
tell = tel.replace("-", "")
if len(tel) == 12 and tell.isdigit() and tel[3] == "-" and tel[7] == "-":
    print("YES")
elif len(tel) == 14 and tell.isdigit() and tel[0] == "7" and tel[1] == "-" and tel[5] == "-" and tel[9] == "-":
    print("YES")
else:
    print("NO")
```

I. Шифровки космонавтов

Степан Павлович Титов был не только наставником, музыкантом, поэтом, художником, но и заядлым эсперантистом («эсперанто» - искусственно созданный международный язык). Язык простой до невозможности: в нём нет исключений из правил, а правил всего 16, в нем 2 падежа и ударение ставится всегда на последний слог.

Титов-старший окончил московские курсы заочного обучения иностранным языкам и вёл переписку с людьми со всего мира. В 1964 году он даже являлся делегатом конгресса эсперантистов в Гааге.

Но на космической орбите уже много лет используется не широко распространенный эсперанто, а другой интернациональный язык: «рунклиш», или «русам», - смесь английского и русского языков. А для передачи сообщений и вовсе алгоритм шифрования. Помогите дешифровальщику получить правильные строки из радиограммы:

1. На вход программе передается строка, состоящая из латинских букв.
2. Одинаковые части строки берутся в скобки, перед скобками записывается число количество повторений.

Напишите программу, которая получает зашифрованную строку и выводит расшифрованную строку.

Формат ввода

Подается зашифрованная строка.

Формат вывода

Программа должна вывести расшифрованную строку.

Пример 1

gfdhhs3(sddsd)fghd	gfdhhssddsdsddsdsddsdfghd

Пример 2

0(s)he0(be)lie0(ve)d	helied

Решение1:

```
string = input()
stack = []
result = ''
count_str = ''
a = 0
for i in range(len(string)):
    if string[i].isdigit():
        a = a * 10 + int(string[i])
        if string[i + 1] == '(':
            if len(stack) >= 1 and type(stack[-1]) == str:
                stack[-1] += count_str
            else:
                stack.append(count_str)
            count_str = ''
    else:
        if string[i] == '(':
            stack.append(a)
        elif string[i] == ')':
            if type(stack[-1]) == int:
                stack[-2] = stack[-2] + count_str * stack[-1]
                stack.pop()
            else:
                stack[-3] = stack[-3] + (stack[-1] + count_str) * stack[-2]
```

```
                stack.pop()
                stack.pop()
            count_str = ''
        else:
            count_str += string[i]
        a = 0
if len(stack) == 0:
    print(count_str)
else:
    print(stack[-1] + count_str)
```

Решение2:

```
s = input()
stack = ''
for ch in s:
    if ch == ')':
        i = len(stack) - 1
        pattern = ''
        while stack[i] != '(':
            pattern = stack[i] + pattern
            i -= 1
        stack = stack[:len(stack) - len(pattern) - 1]
        d = ''
        i = len(stack) - 1
        while i >= 0 and stack[i] in "0123456789":
            d = stack[i] + d
            i -= 1
        stack = stack[:len(stack) - len(d)]
        stack += pattern * int(d)
    else:
        stack += ch
print(stack)
```

J. Ласточкино гнездо

Свой домик в Полковниково Титовы ласково называли «Ласточкино гнездо», сам Герман вспоминал о нем, как о «большом». Удивительно, но эта избенка умудрялась вмещать в себя огромное количество людей – вечерами здесь проходили репетиции, поскольку отец будущего космонавта руководил сельским музыкальным ансамблем.
Чтобы правильно распределить певческие партии в ансамбле, нужно понять, какие голоса сливаются в унисон, а какие нужно разводить. Например, можно выстроить всех участников в ряд и пытаться составлять все возможные подгруппы.
При выделении в группы рекомендуется разбивать только стоящих рядом. Напишите программу, которая поможет руководителю ансамбля сформировать список всех возможных подгрупп.

Формат ввода

На вход программе подается строка текста, содержащая символы, отделенные символом пробела.

Формат вывода

Программа должна вывести указанный список, содержащий все возможные подсписки, включая пустой список в соответствии с примерами.

Пример 1

a b	[[], ['a'], ['b'], ['a', 'b']]

Пример 2

a b v	[[], ['a'], ['b'], ['v'], ['a', 'b'], ['b', 'v'], ['a', 'b', 'v']]

Примечания

Порядок списков одинаковой длины должен соответствовать порядку их вхождения в основной список.
Подсписок — часть другого списка. Подсписок может содержать один элемент, несколько, и даже ни одного. Например, [1], [2], [3] и [4] — подсписки списка [1, 2, 3, 4]. Список [2, 3] — подсписок списка [1, 2, 3, 4], но список [2, 4] не подсписок списка [1, 2, 3, 4], так как элементы 2 и 4 во втором списке не смежные. Пустой список — подсписок любого списка. Сам список — подсписок самого себя, то есть список [1, 2, 3, 4] подсписок списка [1, 2, 3, 4].

Решение1:

```
arr = input().split()
sp = [[]]
k = 1
for i in range(len(arr), 0, -1):
    for j in range(i):
        sp.append(arr[j:j + k])
    k += 1

print(sp)
```

Решение2:

```
from itertools import *

s = input().split()
sp = [[]]
sp = sp + [s[i:j] for i, j in combinations(range(len(s) + 1), 2)]
sp = sorted(sp, key=len)
print(sp)
```

K. Места космической славы

Наследники космонавтики проехали по местам космической славы Клушино, Город Гагарин, Люберцы, Саратов, Оренбург, г. Луостари Мурманской области, Дорогу в космос. Напишите функцию beauty(), которая подберет правильные слова для описания величественного зрелища.

Функция принимает произвольное количество позиционных аргументов – кортежей строк и произвольное количество именованных параметров, среди которых могут быть такие:
duplicate – количество неповторяющихся символов в первой строке кортежа не меньше указанного числа;
second_larger – длина второй строки кортежа больше длины первой строки хотя бы на указанное число;
different_same – первые буквы строк разные, если значение True, и одинаковые, если False, регистр не учитывать;
presence – указанная буква есть в первой строке кортежа (с учетом регистра).
Функция из каждого подходящего кортежа выбирает большую лексикографически без учета регистра строку и возвращает их суммарную длину; а также возвращает самую меньшую лексикографически строку из вторых строк подходящих кортежей.

Формат ввода

В первой строке передается пары слов записанные через точку запятой, в паре используется разделитель запятая и пробел.
Далее вводятся значения именованных аргументов каждое с новой строки (строка может быть пустой).

Формат вывода

Выводит суммарную длину подходящих строк; а также самую меньшую лексикографически строку из вторых строк подходящих кортежей.

Пример 1

attractive, lovely;contemporary, safe;touristic, lively	19 lively

Пример 2

arresting, ravishing;pretty, winsome;alluring, beguiling;Like, impressive;chaRmiNg, lovely 3 1 True l	9 beguiling

Решение1:

```
def beauty(*data1, **conditions1):
    max1, min1 = 0, 'z' * 99
    duplicate1, second_larger1, different_same1, presence1 = None, None, None, None
    if not (conditions1.get('duplicate', None) is None):
        duplicate1 = conditions1['duplicate']
    if not (conditions1.get('second_larger', None) is None):
        second_larger1 = conditions1['second_larger']
    if not (conditions1.get('different_same', None) is None):
        different_same1 = conditions1['different_same']
```

```
    if not (conditions1.get('presence', None) is None):
        presence1 = conditions1['presence']
    for i in data1:
        s1_f = True
        if not (duplicate1 is None):
            if len(set(i[0])) < duplicate1:
                s1_f = False
        if not (second_larger1 is None):
            if len(i[-1]) - len(i[0]) < second_larger1:
                s1_f = False
        if not (different_same1 is None):
            if different_same1:
                if i[0][0].lower() == i[-1][0].lower():
                    s1_f = False
            else:
                if i[0][0].lower() != i[-1][0].lower():
                    s1_f = False
        if not (presence1 is None):
            if presence1 not in i[0]:
                s1_f = False
        if s1_f:
            max1 += len(max(i[0], i[-1], key=lambda x: x.lower()))
            min1 = min(min1, i[-1])
    return max1, min1


data = list(tuple(i.split(", ")) for i in input().split(";"))
conditions = {"duplicate": None,
              "second_larger": None,
              "different_same": None,
              "presence": None}

s1, s2, s3, s4 = input(), input(), input(), input()
if s1 != "":
    conditions["duplicate"] = int(s1)
if s2 != "":
    conditions["second_larger"] = int(s2)
if s3 != '':
    if s3 == "False":
        conditions["different_same"] = False
    else:
        conditions["different_same"] = True
if s4 != "":
    conditions["presence"] = s4
print(*beauty(*data, **conditions))
```

L. Всё, что есть лучшего во мне, – от отца!

На 1 января 2023 года в космосе побывало 597 человек. Из них 72 космонавта СССР, 59 космонавтов России. Но в далеком 1961 году одним из первых людей, кто оказался на околоземной орбите, был наш земляк, Герман Степанович Титов.
Как он стал космонавтом? Кто дал ему дорогу в жизнь, наполненную не только славой, но и тяжелыми тренировками, испытаниями на стойкость и силу характера? Кто был наставником политически грамотного, с нужными для космонавта физическими и медицинскими показателями молодого человека? Космонавт всегда говорил: «Всё, что есть лучшего во мне, – от отца!»

Степан Павлович Титов – отец, учитель, наставник второго космонавта Советского Союза. В школе алтайского села Полковниково он вел начальные классы, преподавал литературу и русский язык, математику и ботанику, черчение и рисование, немецкий язык и пение. Любовь к наукам привил Герману именно он.

– Какие же это науки – пение или игра на инструментах? – Спросите вы.

– Точные! В музыке всё рассчитано до мелочей.

Научитесь и вы анализировать музыкальные фразы. В нашей задаче они записаны не нотами, а рядами целых чисел. Напишите программу, выбирающую из первой строки числа, которых нет во второй и в третьей строках.

Формат ввода

Вводится три строки чисел, записанных через пробел.

Формат вывода

В одну строку через дефис, окруженный пробелами, выведите числа, которые есть в первой строке, но нет во второй и в третьей. Выводить в порядке убывания.

Затем выведите среднее арифметическое таких чисел.

Пример

19 14 17 18 4 17 15 13 19 10 4 5 14 7 5 16 14 5 11 11 19	18 - 17 - 15 16.666666666666668

Примечания

Сортировка, должны быть не в виде чисел, а строк

Решение1:

```
s1 = list(map(int, input().split()))
s2 = list(map(int, input().split()))
s3 = list(map(int, input().split()))
s4 = set(s2) | set(s3)
s1 = set(s1) - set(s4)
print(" - ".join(sorted(list(map(str, s1)), reverse=True)))
print(sum(s1) / len(s1))
```

M. Разложение на простые множители
Требуется разложить целое число N на простые множители с учётом их степени и вывести результат в порядке возрастания множителей.
Формат ввода
Программе дано число N(2≤N≤109)$N(2 \le N \le 109)$.
Формат вывода
Вывести разложение N на простые множители.
Пример 1

2	2

Пример 2

1236	2^2*3*103

Примечания
Возведение в степень обозначайте значком ^.

Решение1:
```
def factorization(n):
    p = []
    d = 2
    while d * d <= n:
        while n % d == 0:
            p.append(d)
            n //= d
        d += 1
    if n > 1:
        p.append(n)
    return p


sl = {}
delit = factorization(int(input()))
for i in delit:
    if i not in sl:
        sl[i] = delit.count(i)
st = ""
for key, value in sl.items():
    if value == 1:
        st += str(key) + "*"
    else:
        st += str(key) + "^" + str(value) + "*"
print(st[:-1])
```

Решение2:
```
import math

number = int(input())

factor = []
for i in range(2, int(math.sqrt(number)) + 1):
    if number % i == 0:
```

```
        cnt = 0
        while number % i == 0:
            number //= i
            cnt += 1
        factor.append((i, cnt))

if number != 1:
    factor.append((number, 1))
print('*'.join(['%d^%d' % i if i[1] > 1 else str(i[0]) for i in factor]))
```

N. Числовые функции

Количество всех натуральных делителей натурального числа n обозначается $\sigma 0(n)$. Сумма всех натуральных делителей числа n обозначается $\sigma 1(n)$.

Формат ввода

Дано натуральное $n \leq 10^9$.

Формат вывода

Выведите σ0(n) и σ1(n).

Пример 1

10	4 18

Пример 2

9	3 13

Примечания

Данную задачу рекомендуется решать путём перебора всех делителей числа до √n.

Решение1:

```
n = int(input())
k = 0
summa_del = 0
r = 0
for i in range(1, int(n ** 0.5) + 1):
    if n % i == 0:
        k += 1
        if i == n // i:
            summa_del += i
            r += 1
        else:
            summa_del += i + (n // i)
print(2 * k - r, summa_del)
```

O. Степень
Для заданного натурального A найти минимальное натуральное N такое, что N в степени N (N, умноженное на себя N раз) делится на A.
Формат ввода
Во входных данных содержится единственное число $1 \le A \le 10^9$.
Формат вывода
Выведите число N.
Пример 1

1	1

Пример 2

9	3

Решение1:

```
def decomp(n):
    ans = []
    d = 2
    while d * d <= n:
        if n % d == 0:
            ans.append(d)
            n //= d
        else:
            d += 1
    if n > 1:
        ans.append(n)
    return ans


x = int(input())
a = list(set(decomp(x)))
b = decomp(x)

y = 1
for i in range(len(a)):
    y *= a[i]
k = 1
n = k * y

if x == 1:
    print(1)
elif len(b) > 29:
    print(y)
else:
    while pow(n, n, x) != 0:
        n = k * y
        k += 1
    print(n)
```

Решение2:

```python
from math import gcd

a = int(input())
for n in range(1, 10 * 5):
    a1 = a
    for k in range(n):
        a1 //= gcd(n, a1)
    if a1 == 1:
        break
if a1 != 1:
    n = 1
    p = 2
    while p * p <= a:
        if a % p == 0:
            n *= p
            while a % p == 0:
                a //= p
        p += 1
    if a > 1:
        n *= a
print(n)
```

P. Секретное слово

Напишите программу для расшифровки секретного слова методом частотного анализа.

Формат ввода

В первой строке задано зашифрованное слово. Во второй строке задано одно целое число n – количество букв в словаре. В следующих n строках записано, сколько раз конкретная буква алфавита встречается в этом слове – <буква>: <частота>.

Формат вывода

Программа должна вывести дешифрованное слово.

Пример

pop 2 д: 2 e: 1	дед

Примечания

Гарантируется, что частоты букв не повторяются.

Решение1:

```
stroka = input()
n = int(input())
sl = {}
for i in range(n):
    value, key = input().split(":")
    sl[int(key)] = value
new_stroka = ""
for i in stroka:
    new_stroka += sl[stroka.count(i)]
print(new_stroka)
```

Решение2:

```
s = input()
n = int(input())
for i in range(n):
    x = (input().split(": "))
    for j in s:
        if s.count(j) == int(x[1]):
            s = s.replace(j, x[0])
print(s)
```

Q. Пароли

Пароль от некоторого сервиса должен удовлетворять таким ограничениям:
- состоять из символов таблицы ASCII с кодами от 33 до 126;
- быть не короче 8 символов и не длиннее 14;
- из 4 классов символов — большие буквы, маленькие буквы, цифры, прочие символы — в пароле должны присутствовать не менее трёх любых.

Напишите программу, которая проверит, что введённый пароль подходит под эти ограничения.

Формат ввода

На входе дана одна строка с паролем.

Формат вывода

Выведите YES, если пароль удовлетворяет требованиям, и NO в противном случае.

Пример

Vasya123	YES

Решение1:

```
pas = input()
a = any([p.isdigit() for p in pas])
b = any([p.isupper() for p in pas])
c = any([p.islower() for p in pas])
e = all([33 <= ord(p) <= 126 for p in pas])
if 14 >= len(pas) >= 8 and e and ((a and b and c) or (a and b and not c) or (a and c and not b) or (c and b and not a)):
    print('YES')
else:
    print('NO')
```

R. Искусственный интеллект и шахматы

Компьютерные программы на основе "искусственного интеллекта" для игры в шахматы показывают улучшение результатов каждый год. Для этого им требуется быстро и качественно распознавать различные ситуации, возникающие на шахматной доске. Вам требуется написать программу, которая для заданных "матовых" позиций на доске определит, поставлен ли мат с помощью ладьи или ферзя.

Формат ввода

В первой строке одно целое число N <= 10 - количество позиций. Далее описывается N шахматных позиций, по 8 строк на каждое описание. В каждой строке через пробел записаны коды фигур, стоящих на поле, либо цифра 0, если в клетке фигуры нет. Описание фигуры состоит из двух латинских символов: первый - буква w или b, описывающая цвет фигуры - белый или чёрный. Второй символ - код фигуры: король - K, ферзь - Q, ладья - R, конь - N, слон - B, пешка - p.

Формат вывода

Одно число - количество позиций среди заданных, где король находится под шахом ладьи или ферзя.

Пример

1 wR 0 0 0 0 0 0 bK 0 0 0 wR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1

Примечания

Гарантируется, что на каждой позиции поставлен мат белым или чёрным.

Решение1:

```
n = int(input())
color = "wb"
c = 0
for i in range(n):
    field = []
    for j in range(8):
        field.append(input().split())

    f = False
    for j in range(8):
        for k in range(8):
            if field[j][k][-1] == 'K':
                oc = color[(color.index(field[j][k][0]) + 1) % 2]
                for l in range(8):
                    if (field[j][l] == oc + "R") or (field[j][l] == oc + "Q"):
                        f = True
                        break
                for l in range(8):
                    if (field[l][k] == oc + "R") or (field[l][k] == oc + "Q"):
                        f = True
                        break
```

```
            for l in range(1, 8):
                if ((j - l) >= 0 and (k - l) > 0 and (field[j - l][k - l] == oc + "Q")) or \
                        ((j + l < 8) and (k + l < 8) and (field[j + l][k + l] == oc + "Q")) or \
                        ((j - l >= 0) and (k + l < 8) and (field[j - l][k + l] == oc + "Q")) or \
                        ((j + l < 8) and (k - l >= 0) and (field[j + l][k - l] == oc + "Q")):
                    f = True
                    break
    if f:
        c += 1

print(c)
```

Решение2:

```
n = int(input())
masf = []
kol = 0
mas_w = [] * 5
mas_b = [] * 5
for h in range(n):
    mas = [[str(j) for j in (input().split())] for i in range(8)]
    for i in range(8):
        for j in range(8):
            if mas[i][j] == 'bK':
                masf = []
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_w.append(masf)
            elif mas[i][j] == 'wK':
                masf = []
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_b.append(masf)
    for i in range(8):
        for j in range(8):
            if mas[i][j] == 'wR' or mas[i][j] == 'wQ':
                masf = []
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_w.append(masf)
            elif mas[i][j] == 'bR' or mas[i][j] == 'bQ':
                masf = []
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_b.append(masf)
```

```
            if len(mas_w) > 1:
                f = False
                pos_k_x = mas_w[0][1]
                pos_k_y = mas_w[0][2]
        for i in range(1, len(mas_w)):
            if not (f) and mas_w[i][1] == pos_k_x or mas_w[i][2] == pos_k_y or (
                    (mas_w[i][0] == 'bQ') and (abs(pos_k_x - mas_w[i][1]) == abs(pos_k_y - mas_w[i][2]))):
                kol += 1
                f = True
        if len(mas_b) > 1:
            f = False
            pos_k_x = mas_b[0][1]
            pos_k_y = mas_b[0][2]
        for i in range(1, len(mas_b)):
            if not (f) and mas_b[i][1] == pos_k_x or mas_b[i][2] == pos_k_y or (
                    (mas_b[i][0] == 'wQ') and (abs(pos_k_x - mas_b[i][1]) == abs(pos_k_y - mas_b[i][2]))):
                kol += 1
                f = True
        mas_w = [] * 5
        mas_b = [] * 5
print(kol)
```

S. Поиск Фибоначчи

В квадратной матрице NxN требуется найти последовательность чисел, удовлетворяющую определённому соотношению. Числа должны располагаться под побочной диагональю или лежать на ней и при просмотре по спирали против часовой стрелки, начиная с правого верхнего угла, образовывать последовательность Фибоначчи. Диагональ, идущая в квадратной матрице из левого нижнего угла в правый верхний, называется побочной. Числа Фибоначчи - элементы числовой последовательности, первые два числа которой равны 0 и 1, а каждое последующее - сумма двух предыдущих: 0, 1, 1, 2, 3, 5...

Формат ввода

В первой строке задано единственное натуральное число N <= 100 - порядок матрицы. Далее идёт N строк по N целых чисел, разделённых одним или несколькими пробелами и описывающими элементы матрицы построчно. Каждый из элементов матрицы по модулю не превышает 10^6.

Формат вывода

Найденные элементы последовательности Фибоначчи, удовлетворяющие условию, записанные в строку через пробел.

Пример

4 0 1 2 3 4 5 0 3 7 8 5 2 1 1 2 3	0 1 1 2 3 5

Решение1:

```
from math import ceil

n = int(input())
m = []
for j in range(n):
    mm = list(map(int, input().split()))[n - 1 - j:]
    m.append(mm)
t = [m[0][0]]
for k in range(ceil(n // 2)):
    for j in range(k + 1, n - k):
        t.append(m[j][k])
    for j in range(k + 1, n - 2 * k):
        t.append(m[- 1 - k][j])
    for j in range(n - 2 * k - 2, k, -1):
        t.append(m[j][-1 - k])
fb = []
for i in range(len(t)):
    if len(fb) == 0:
        if t[i] == 0:
            fb.append(0)
    elif len(fb) == 1:
        if t[i] == 1:
            fb.append(1)
    elif len(fb) >= 2:
        if t[i] == fb[-1] + fb[-2]:
```

```
            fb.append((t[i]))
print(*fb)
```

Решение2:

```
n = int(input())
m = [0] * n
fib = [0]
for i in range(n):
    m[i] = list(map(int, input().split()))
nt = n
t = 0
mn = []
i1 = 0
t = 0
while True:
    j1 = nt - 1
    if nt == 0: break
    for i in range(i1, nt):
        mn.append(m[i][j1])
        if m[i][j1] == fib[-1]:
            if fib[-1] == 0:
                fib.append(1)
            else:
                fib.append(fib[-1] + fib[-2])
        j1 -= 1
    for j in range(j1 + 1, nt):
        mn.append(m[nt - 1][j])
        if m[nt - 1][j] == fib[-1]: fib.append(fib[-1] + fib[-2])
    for i in range(nt - 1, i1, -1):
        mn.append(m[i][nt - 1])
        if m[i][nt - 1] == fib[-1]: fib.append(fib[-1] + fib[-2])
    i1 += 2
    t += 1
    nt -= 1
fib.pop()
for a in fib: print(a, end=' ')
```

T. Кубики

Строителям привезли N кубиков из которых необходимо построить лестницу. Перед строительством лестницы прораб решил предложить заказчику различные варианты лестниц. После отрисовки макетов прораб задумался, а все ли варианты он нарисовал. Помогите прорабу проверить все ли лестницы он нарисовал. Напишите программу, которая посчитает число лестниц, которые можно построить из N кубиков.

Формат ввода

Во входном файле INPUT.TXT записано натуральное число N (1 ≤ N ≤ 100) – количество кубиков в лесенке.

Формат вывода

В выходной файл OUTPUT.TXT необходимо вывести число лесенок, которые можно построить из N кубиков.

Пример 1

6	4

Пример 2

5	3

Примечания

Лестницей называется набор кубиков, в котором каждый более верхний слой содержит меньше кубиков, чем предыдущий.

Решение1:

```
def f(k, n):
    kol = 0
    if n == 0:
        return kol + 1
    else:
        if k < n:
            for i in range(k + 1, n + 1):
                kol = kol + f(i, n - i)
            return kol
        else:
            return kol


f1 = open("input.txt")
n = int(f1.readline())
f2 = open("output.txt", "w")
f2.write('{}'.format(f(0, n)))
f2.close()
```

U. Задача4_олимп

Фермер решил разделить n-угольный участок между двумя сыновьями. Деление должно осуществляться таким образом, чтобы линия деления проходила через одну из сторон участка. Помогите Фермеру определить, можно ли произвести такое деление.

Формат ввода

В первой строке вводится одно число N (3≤N≤100000). Далее в N строках задается по паре чисел (x, y) местоположение столбов в изгороди.

Формат вывода

Выведите одну строку: “Можно”, если приведённый участок возможно разделить по стороне на две части, и “Нельзя” в противном случае.

Пример 1

8 1 0 0 1 0 2 1 3 2 3 3 2 3 1 2 0	Нельзя

Пример 2

12 0 0 0 1 -1 1 -1 2 2 2 2 1 1 1 1 0 2 0 2 -1 -1 -1 -1 0	Можно

Примечания

Пары чисел могут задаваться через пробел или табуляцию

Решение1:

```
class Dot:
    def __init__(self, line):
        self.x, self.y = map(int, line.split())


def check(A, B, C, D):
    a = C.y - B.y
    b = B.x - C.x
    c = C.x * B.y - C.y * B.x
    return (a * A.x + b * A.y + c) * (a * D.x + b * D.y + c) > 0


def main():
```

```
    n = int(input())
    if n == 3:
        print("Нельзя")
        return
    a = [Dot(input()) for _ in range(n)]
    for i in range(n):
        if not check(a[i], a[i - 1], a[i - 2], a[i - 3]):
            print("Можно")
            break
    else:
        print("Нельзя")


main()
```

V. Задача13_олимп

Почти все дети очень любят такую игру: забраться на пенёк и перепрыгнуть с него на другой. А если таких пеньков много, то можно упрыгать довольно далеко, не слезая на землю. А если получится перепрыгивать через пенёк или через несколько, то становится ещё веселее!

Давайте высоту выстроенных в ряд пеньков обозначим просто целыми числами. Перепрыгивать интереснее всего не на соседний пенёк, а с самого высокого (локально) на другой самый высокий в своём окружении. Пеньки, высота которых больше соседей справа и слева, будем называть локальными максимумами. Тогда первый и последний такими максимумами быть не могут — у них нет одного соседа.

Напишите программу, которая в последовательности целых чисел, заканчивающейся числом 0 (в последовательность не входит, признак ее окончания), находит наибольшее расстояние между двумя локальными максимумами.

Формат ввода

Последовательность целых чисел, каждое с новой строки, заканчивается числом 0.

Формат вывода

Максимальное расстояние между соседними локальными максимумами. Расстоянием считается разность номеров элементов последовательности. Например, расстояние между 2 и 5 элементами равно 3.

Пример 1

1 2 3 4 53 434 3 34 5 7 8 0	2

Пример 2

1 3 5 4 2 1 5 3 1 5 2 3 0	4

Решение1:

```
ch = int(input())
sp = []
while ch != 0:
    sp.append(ch)
```

```
    ch = int(input())
sp1 = []
for i in range(1, len(sp) - 1):
    if sp[i - 1] < sp[i] > sp[i + 1]:
        sp1.append(i)
if len(sp1) < 2:
    print(len(sp))
else:
    loc_max = sp1[1] - sp1[0]
    for i in range(2, len(sp1)):
        if sp1[i] - sp1[i - 1] > loc_max:
            loc_max = sp1[i] - sp1[i - 1]
    print(loc_max)
```

W. Задача15_олимп

Маской называется строка, состоящая из английских букв (a, ..., z, A, ..., Z) и символов ? и *. Каждый из символов ? разрешается заменить на одну произвольную букву, а каждый из символов * – на произвольную (возможно пустую) последовательность букв. Про любую строку из букв, которую можно получить из маски такими заменами, будем говорить, что она удовлетворяет этой маске.

Имеются две маски. Требуется найти строку минимальной длины, которая удовлетворяет обоим маскам, либо выдать сообщение, что такой строки не существует.

Формат ввода

Заданные маски записаны в первых двух строках входных данных. Длина каждой маски не превосходит 80 символов.

Формат вывода

Выведите длину подходящей строки, удовлетворяющую обоим маскам, либо сообщение "No".

Пример 1

AB CD	4

Пример 2

AB? *BC	3

Пример 3

??? *	3

Решение1:

```
def matched_string(first_regexp: str, second_regexp: str) -> str:
    DEFAULT = 'a' * 300
    d = [[DEFAULT for _ in range(1000)] for _ in range(1000)]
    d[0][0] = ''

    if first_regexp[0] == '*':
        d[1][0] = ''

    if second_regexp[0] == '*':
        d[0][1] = ''

    for i in range(1, len(first_regexp) + 1):
        for j in range(1, len(second_regexp) + 1):
            check_first = d[i - 1][j]
            check_second = d[i][j - 1]
            if first_regexp[i - 1] == '*':
                if check_first == DEFAULT and check_second == DEFAULT:
                    d[i][j] = DEFAULT
                elif second_regexp[j - 1] == '*':
                    d[i][j] = min(check_first, check_second, key=len)
                elif second_regexp[j - 1] == '?':
                    d[i][j] = min(check_first, check_second + 'a', key=len)
                else:
                    d[i][j] = min(check_first, check_second + second_regexp[j - 1], key=len)
            elif second_regexp[j - 1] == '*':
```

```
                if check_first == DEFAULT and check_second == DEFAULT:
                    d[i][j] = DEFAULT
                elif first_regexp[i - 1] == '?':
                    d[i][j] = min(check_second, check_first + 'a', key=len)
                else:
                    d[i][j] = min(check_second, check_first + first_regexp[i - 1], key=len)
            elif d[i - 1][j - 1] == DEFAULT:
                d[i][j] = DEFAULT
            elif first_regexp[i - 1] == '?':
                if second_regexp[j - 1] == '?':
                    d[i][j] = d[i - 1][j - 1] + 'a'
                else:
                    d[i][j] = d[i - 1][j - 1] + second_regexp[j - 1]
            elif second_regexp[j - 1] == '?':
                d[i][j] = d[i - 1][j - 1] + first_regexp[i - 1]
            else:
                if first_regexp[i - 1] == second_regexp[j - 1]:
                    d[i][j] = d[i - 1][j - 1] + first_regexp[i - 1]
                else:
                    d[i][j] = DEFAULT
    ans = d[len(first_regexp)][len(second_regexp)]
    if ans == DEFAULT:
        ans = "No"
    return ans


if __name__ == '__main__':
    first_regexp = input()
    second_regexp = input()
    res = matched_string(first_regexp, second_regexp)
    if res != "No":
        print(len(res))
    else:
        print(res)
```

X. Задача16_олимп

Фермер научился путешествовать в параллельный мир, который очень похож на наш, однако в нем все объекты отражены горизонтально. Все, что попадает в этот мир из реального, тоже «отзеркаливается», но фермер изобрел специальные ящики, содержимое которых остается в изначальном состоянии после переноса в новый мир.

Ящики выглядят как прямоугольники разных размеров, у которых нижняя и верхняя части обозначены символом _, а стенки — символом |. Содержимое ящиков состоит из букв латинского алфавита в произвольном регистре и цифр. Все пустое пространство на складе составляют символы .. Стенки, нижние и верхние части у ящиков могут совпадать.

Напишите программу, которая показывает, как будут выглядеть ящики после их переноса в зеркальное измерение.

Формат ввода

В первой строке программе подается два числа nn и mm — высота и ширина склада с ящиками соответственно, где 1≤n,m≤99. В последующих n строках длины m изображается склад с ящиками, структура которого описана в условии задачи.

Формат вывода

Программа должна вывести n строк длины m — состояние склада после его переноса в зеркальный мир с отраженными по горизонтали ящиками, содержимое которых отражено не было.

Пример 1

4 4 .__. .|a| |12|	.__. |a|. |12| ....

Пример 2

4 7 .__.__. .|a|... |aS|_.. ....|s|	.__.__. ...|a|. .._|aS| |s|....

Решение1:

```
n, m = map(int, input().split())

symbol = "0123456789qwertyuioplkjhgfdsazxcvbnmQWERTYUIOPLKJHGFDSAZXCVBNM"

for i in range(n):
    word = input()
    line = ""
    line1 = ""
    j = 0
    while j < m:
        if word[j] not in symbol:
            line = word[j] + line
            j += 1
        else:
            while word[j] in symbol:
                line1 += word[j]
                j += 1
            line = line1 + line
```

```
        line1 = ""
print(line)
```