Вариант олимпиады по ИИ для 9–11 классов

Задача 1. Шифр ДНК

Учёные из школьного биолаборатория изучают фрагмент ДНК, представленный строкой из букв `A, C, G, T`. Для исследования важно посчитать, сколько различных последовательностей длины `k` встречается в цепочке.

Формат ввода

Строка (длина ≤ 10^5), число k.

Формат вывода

Одно целое — количество уникальных подстрок.

Пример

Ввод

```
ACGTACG
3
```

Вывод

```
4
```

Тесты

```
Ввод:
AAAAA
2
Вывод:
1

Ввод:
ACGT
5
Вывод:
0

Ввод:
ACACAC
2
Вывод:
2

Ввод:
G
1
Вывод:
1
```

Решение задачи 1

```
def main():
    s = input().strip()
    k = int(input())
    n = len(s)
    if k > n or k <= 0:
        print(0)
        return
    seen = set()
    for i in range(n - k + 1):
        seen.add(s[i:i+k])
    print(len(seen))


if __name__ == "__main__":
    main()
```

Задача 2. Вероятность победы

В школьном клубе проходит турнир роботов. Каждый матч длится до двух побед. Наш робот выигрывает отдельную партию с вероятностью `p = a/b`. Надо вычислить вероятность того, что робот выиграет весь матч. Ответ нужно вывести в виде несократимой дроби.

Формат ввода

Два целых `a b` (p = a/b).

Формат вывода

Несократимая дробь `num den`.

Пример

Ввод

```
1 3
```

Вывод

```
7 27
```

Тесты

```
Ввод:
1 1
Вывод:
1 1

Ввод:
1 2
Вывод:
1 2

Ввод:
2 3
Вывод:
20 27

Ввод:
2 5
Вывод:
44 125
```

Решение задачи 2

```
import math


def main():
    a, b = map(int, input().split())
    num = a*a * (3*b - 2*a)
    den = b*b*b
    g = math.gcd(num, den)
    print(num // g, den // g)


if __name__ == "__main__":
    main()
```

Задача 3. Рекомендации фильмов

В школьном киноклубе каждый ученик хранит список любимых фильмов. Когда приходит новичок, нужно порекомендовать ему фильмы на основе вкусов других.

- Вес ученика = количество общих фильмов с новичком.
- Оценка фильма = сумма весов учеников, у которых этот фильм есть.
- Рекомендуем только фильмы, которых у новичка нет.
- При равенстве сортируем фильмы по алфавиту.

Составьте рекомендации — первые K фильмов по этим правилам.

Формат ввода

1. Целое `N` (1 ≤ N ≤ 200).
2. N строк: списки фильмов учеников, через запятую.
3. Строка со списком фильмов новичка.
4. Целое `K` (1 ≤ K ≤ 10).

Формат вывода

Строка: K фильмов через запятую. Если фильмов нет — выведите `-`.

Пример

Ввод

```
4
avatar,lotr,shrek
lotr,hobbit,harrypotter
shrek,bee_movie
avatar,hobbit
avatar,shrek
3
```

Вывод

```
lotr,bee_movie,hobbit
```

Тесты

```
Ввод:
2
a,b
c,d
a
2
Вывод:
b

Ввод:
2
a,b
a,c
a
2
```

```
Вывод:
b,c

Ввод:
2
a,b
b,c
a,b
2
Вывод:
c

Ввод:
1
a,b,c
a,b,c
2
Вывод:
-
```

Решение задачи 3

```
def parse_list(line: str):
    items = [x.strip() for x in line.split(',') if x.strip() != '']
    return set(items)

def main():
    N = int(input())
    students = [parse_list(input().strip()) for _ in range(N)]
    newbie = parse_list(input().strip())
    K = int(input())

    weights = []
    for st in students:
        w = len(st & newbie)
        weights.append(w)
```

```
    score = {}
    for st, w in zip(students, weights):
        if w == 0:
            continue
        for f in st:
            if f not in newbie:
                score[f] = score.get(f, 0) + w

    if not score:
        print('-')
        return

    ranked = sorted(score.items(), key=lambda x: (-x[1], x[0]))
    recs = [name for name, _ in ranked[:K]]
    print(','.join(recs))

if __name__ == "__main__":
    main()
```

Задача 4. Кластеризация точек

Экспедиция нанесла на карту n точек с координатами. Нужно разделить их на два лагеря:

- Лагерь «Север» — ближе к (0,0),
- Лагерь «Юг» — ближе к (1000,1000).
- При равенстве точка идёт в лагерь «Север».

Разделите все точки и выведите индексы участников по лагерям в порядке их появления.

Формат ввода

n ($1 \le n \le 2 \cdot 10^5$), далее n пар координат.

Формат вывода

Две строки — индексы точек каждого кластера в порядке ввода.

Пример

Ввод

```
3
1 1
500 500
999 999
```

Вывод

```
1 2
3
```

Тесты

```
Ввод:
2
0 0
1000 1000
Вывод:
1
2

Ввод:
1
1000 0
Вывод:
1

Ввод:
1
0 1000
Вывод:
1

Ввод:
2
1000 0
1001 1001
Вывод:
1
2
```

Решение задачи 4

```
def main():
    n = int(input())
    north = []
```

```
    south = []
    for idx in range(1, n+1):
        x, y = map(int, input().split())
        d0 = x*x + y*y
        dx = x - 1000
        dy = y - 1000
        d1 = dx*dx + dy*dy
        if d0 <= d1:
            north.append(idx)
        else:
            south.append(idx)
    print(' '.join(map(str, north)))
    print(' '.join(map(str, south)))

if __name__ == "__main__":
    main()
```

Задача 5. Симуляция очереди

В школьном буфете очередь. Каждое событие описано командой:

- `+ имя` — ученик пришёл в конец очереди,
- – первый ученик получил еду и ушёл.

Смоделируйте очередь и выведите имена в порядке обслуживания.

Формат ввода

m (≤ 10^5), далее m команд.

Формат вывода

Все имена по одному в строке.

Пример

Ввод

```
5
+ anna
+ bob
-
+ clara
-
```

Вывод

```
anna
bob
```

Тесты

Ввод:
```
3
+ a
+ b
-
```
Вывод:
```
a
```

Ввод:
```
4
+ a
+ b
-
-
```
Вывод:
```
a
b
```

Ввод:
```
2
+ x
-
```
Вывод:
```
x
```

Ввод:
```
6
+ a
+ b
+ c
-
-
-
```
Вывод:
```
a
b
c
```

```
# Решение задачи 5
```

```
from collections import deque


def main():
    m = int(input())
    q = deque()
    out = []
    for _ in range(m):
        line = input().strip()
        if line[0] == '+':
            name = line[1:].strip()
            q.append(name)
        else:
            out.append(q.popleft())
    print('\n'.join(out))


if __name__ == "__main__":
    main()
```

Задача 6. Перемножение матриц

В школе внедряют систему умных табло, которая должна умножать матрицы: матрица расписания $A(n \times m)$ умножается на матрицу коэффициентов занятости $B(m \times k)$. Вычислите произведение $A \cdot B$.

Формат ввода

n, m, k (≤ 200), затем A и B.

Формат вывода

Матрица n×k.

Пример

Ввод

```
2 2 2
1 2
3 4
5 6
7 8
```

Вывод

```
19 22
43 50
```

Тесты

```
Ввод:
1 1 1
5
7
Вывод:
35
```

```
Ввод:
2 1 2
3
4
5 6
Вывод:
15 18
20 24
```

```
Ввод:
1 2 1
1 2
3
4
Вывод:
11
```

```
Ввод:
3 1 1
1
2
3
4
Вывод:
4
8
12
```

Решение задачи 6

```
def main():
    n, m, k = map(int, input().split())
    A = [list(map(int, input().split())) for _ in range(n)]
    B = [list(map(int, input().split())) for _ in range(m)]

    BT = [[B[r][c] for r in range(m)] for c in range(k)]

    C = []
    for i in range(n):
        row = []
        Ai = A[i]
        for c in range(k):
            s = 0
            Bc = BT[c]
            for j in range(m):
                s += Ai[j] * Bc[j]
            row.append(s)
        C.append(row)

    for row in C:
        print(' '.join(map(str, row)))

if __name__ == "__main__":
    main()
```

Задача 7. Монте-Карло интеграл

Учитель информатики показал метод Монте-Карло на примере игры. На квадрате $[0,1] \times [0,1]$ случайно выбираются точки. Если точка лежит под графиком $f(x) = x^2$, то она «поймана». По заданным точкам найдите оценку интеграла $\int_0^1 x^2\,dx$. Ответ округлите до трёх знаков.

Формат ввода

N и последовательность 2N чисел (координаты).

Формат вывода

Оценка интеграла (3 знака).

Пример

Ввод

```
5
0.1 0.01 0.3 0.2 0.5 0.4 0.7 0.6 0.9 0.7
```

Вывод

```
0.400
```

Тесты

```
Ввод:
1
0.5 0.1
Вывод:
1.000
```

```
Ввод:
1
0.5 0.3
Вывод:
0.000
```

```
Ввод:
2
0.0 0.0 1.0 1.0
Вывод:
1.000
```

```
Ввод:
2
0.5 0.1  0.5 0.5
Вывод:
0.500
```

Решение задачи 7

```
def main():
    N = int(input())
    coords = list(map(float, input().split()))
    caught = 0
```

```
    for i in range(0, 2*N, 2):
        x, y = coords[i], coords[i+1]
        if y <= x*x:
            caught += 1
    estimate = caught / N if N > 0 else 0.0
    print(f"{estimate:.3f}")


if __name__ == "__main__":
    main()
```

Задача 8. Нейросеть для XOR

Учитель робототехники рассказал школьникам, что не все логические функции можно реализовать одним линейным классификатором. Например, функция **XOR** (исключающее «ИЛИ») принимает два бинарных входа ($x, y \in 0,1$) и возвращает 1, если ровно один из входов равен 1, иначе 0:
- $(x = 0, y = 0; \Rightarrow; 0)$.
- $(x = 1, y = 0; \Rightarrow; 1)$.
- $(x = 0, y = 1; \Rightarrow; 1)$.
- $(x = 1, y = 1; \Rightarrow; 0)$.

Оказалось, что для решения этой задачи нужно построить **маленькую нейросеть** с одним скрытым слоем из двух нейронов и одним выходным с одним нейроном (считаем, что функция активации нейрона пороговая).
Напишите программу на одном из предложенных языков программирования, которая реализует нейросеть на Python, которая обучается на таблице истинности XOR и правильно предсказывает результат для всех четырёх входных комбинаций. Напишите все параметры нейронной сети после обучения:
- параметры первого нейрона скрытого слоя w11, w12, b1,
- параметры второго нейрона скрытого слоя w21, w22, b2,
- параметры нейрона выходного слоя W1, W2, B.

Формат ввода
Ввод отсутствует (таблица истинности зашита в программу).
Формат вывода
0.5 0.5 0.5
0.5 0.5 0.5
0.5 0.5 0.5

```
# Решение задачи 8


def check_xor_network():
```

```
# Читаем параметры
hidden1 = list(map(float, input().split()))
hidden2 = list(map(float, input().split()))
output = list(map(float, input().split()))

# Проверяем корректность ввода
if len(hidden1) != 3 or len(hidden2) != 3 or len(output) != 3:
    return False

# Распаковываем параметры
w11, w12, b1 = hidden1  # Нейрон h₁
w21, w22, b2 = hidden2  # Нейрон h₂
w31, w32, b3 = output   # Нейрон y

# Пороговая функция активации
def step_function(x):
    return 1 if x >= 0 else 0

def forward(x1, x2):
    """Прямое распространение для входов x1, x2"""
    # Скрытый слой
    h1 = step_function(w11 * x1 + w12 * x2 + b1)
    h2 = step_function(w21 * x1 + w22 * x2 + b2)

    # Выходной слой
    y = step_function(w31 * h1 + w32 * h2 + b3)

    return y
```

```
# Тестовые данные XOR
test_cases = [
    (0, 0, 0),
    (0, 1, 1),
    (1, 0, 1),
    (1, 1, 0)
]

# Проверяем все тестовые случаи
for x1, x2, expected in test_cases:
    if forward(x1, x2) != expected:
        return False

return True
```