



Заключительный этап Олимпиады школьников «Шаг в будущее»

Профиль: «Инженерное дело»

Специализация: «Информатика»

Классы участия: 10-11

Задача 1

Условие

В маленьком портовом городке каждую весну проходит шумная ярмарка. Жители окрестных деревень собираются, чтобы посмотреть на яркие представления, отведать местных деликатесов и поторговаться на главной площади. Единственная железнодорожная ветка, ведущая в этот город, обслуживается старым поездом с особыми правилами посадки.

Дело в том, что в каждом вагоне этого поезда расположено **4 купе**, а в каждом купе ровно **4 места**. Однако каждое из этих мест рассчитано на пассажиров определённого роста:

- **1-е место:** для пассажиров ростом **до 150 см** (включительно);
- **2-е место:** для пассажиров ростом **от 151 до 180 см** (включительно);
- **3-е место:** для пассажиров ростом **от 181 до 200 см** (включительно);
- **4-е место:** для пассажиров ростом **от 201 до 230 см** (включительно).

Чтобы не возникало путаницы, существует правило: пассажира всегда стараются посадить на **наименьшее** подходящее по ограничению место. Например, если пассажир ростом 145 см может сесть на место 1-е место (до 150 см) или на второе место (до 180 см), его обязательно посадят на 1-е место (до 150 см).

Когда на вокзал приезжает толпа людей, каждому нужно найти подходящее свободное место. Если в текущих вагонах места закончились, к поезду прицепляют новый вагон. Таким образом, стоит важная задача — определить, **сколько всего вагонов** понадобится прицепить к поезду, чтобы все пассажиры смогли занять места согласно указанным правилам.

Входные данные

В первой строке задано натуральное число N ($1 \leq N \leq 10^5$) — количество пассажиров, прибывших на вокзал.



В следующих N строках записаны целые числа — рост каждого пассажира (по одному числу в строке в диапазоне от 130 до 230 включительно).

Выходные данные

Вывести в качестве ответа одно целое число — минимальное количество вагонов, необходимых для размещения всех пассажиров.

Входные данные	Выходные данные
5 140 160 180 195 220	1
8 150 230 230 230 150 190 230 230	2

Пояснение к примеру #1:

В одном вагоне есть 4 купе по 4 места, итого $4 \cdot 4 = 16$ мест. Каждое место имеет своё ограничение по росту. Для пяти пассажиров достаточно одного вагона.

- Пассажир ростом 140 см займёт место с ограничением до 150 см.
- Пассажир ростом 160 см и 180 см — 2 места с ограничением до 180 см.
- Пассажир ростом 195 см — место с ограничением до 200 см.
- Пассажир ростом 220 см — место с ограничением до 230 см.



Таким образом, все они легко размещаются в одном вагоне, и дополнительных вагонов не требуется.

Пример решения

```
N = int(input())
pers = []
for i in range(N):
    pers.append(int(input()))

vags = []

pers.sort(reverse=True)

for p in pers:
    flag_find = False
    for v in vags:
        for kupe in v:
            if p > 200:
                if kupe[3] == 0:
                    kupe[3] = 1
                    flag_find = True
                    break
            elif p > 180:
                for j in range(3, 1, -1):
                    if kupe[j] == 0:
                        kupe[j] = 1
                        flag_find = True
                        break
            elif p > 150:
                for j in range(3, 0, -1):
                    if kupe[j] == 0:
                        kupe[j] = 1
                        flag_find = True
                        break
            else:
                for j in range(3, -1, -1):
                    if kupe[j] == 0:
                        kupe[j] = 1
                        flag_find = True
                        break
        if flag_find:
            break
    if flag_find:
        break

if not flag_find:
    vags += [
        [0, 0, 0, 1],
        [0, 0, 0, 0],
        [0, 0, 0, 0],
        [0, 0, 0, 0]
```



Федеральное государственное автономное
образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана (национальный исследовательский университет)»

ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

```
]]  
print(len(vags))
```

Критерии оценивания**

<i>Критерий</i>	<i>Балл</i>
Дан неверный ответ/ответ отсутствует	0
Дан верный ответ	5

** - Балл за каждую задачу проставляется в соответствии с долей выполнения участником задания. Шаг оценивания каждого задания - 1 балл. Доля выполнения вычисляется автоматической системой тестирования на основе комплекта установленных для задания тестов.



Задача 2

Условие

На борту орбитальной космической станции готовится торжественная фотосессия. Все космонавты собрались в большом прямоугольном модуле со стеклянной стеной, чтобы сделать снимок на фоне Земли. Однако у каждого космонавта скафандр имеет свою ширину, поэтому не все могут поместиться в кадр.

Для того, чтобы все космонавты поместились в кадр, решили определить первоначально минимальную ширину кадра, которую могут занять космонавты, чтобы все попали в кадр, именно поэтому сформировали ряд правил:

- космонавты могут на кадре стоять вплотную друг к другу (между шириной скафандра одного не будет расстояния до скафандра другого);
- космонавты могут вставать в несколько рядов (друг за другом), но космонавт может встать за другим только в том случае, если он выше всех стоящих в ряду перед ним, минимум на 20 сантиметров.

Входные данные

На первой строке подаётся натуральное число N (от 2 до 6, включительно) – количество космонавтов.

Далее на N строках подаётся по два целых числа через пробел: W (от 40 до 120) в сантиметрах – ширина скафандра, H (от 160 до 230) в сантиметрах – высота космонавта.

Выходные данные

В первой строке вывести минимальную ширину, которую могут занять космонавты перед съемкой.



Входные данные	Выходные данные
4 40 160 50 160 60 200 45 210	105
3 40 160 120 170 50 180	210

Пример решения

```
N = int(input())
pers = []
for i in range(N):
    pers.append(tuple(map(int, input().split())))
pers.sort(key=lambda x: (x[1], x[0]))
rows = [[0, 0]]
for p in pers:
    w, h = p
    if h - rows[-1][1] >= 20:
        rows.append([w, h])
    else:
        rows[-1][0] += w
        rows[-1][1] = h
max_w = 0
for r in rows:
    max_w = max(max_w, r[0])
print(max_w)
```

Критерии оценивания**

Критерий	Балл
Дан неверный ответ/ответ отсутствует	0
Дан верный ответ	8

** - Балл за каждую задачу проставляется в соответствии с долей выполнения участником задания. Шаг оценивания каждого задания - 1 балл. Доля выполнения вычисляется автоматической системой тестирования на основе комплекта установленных для задания тестов.



Задача 3

Условие

В крупном IT-офисе есть ровно две переговорные комнаты — №1 и №2.

Сотрудники за день присылают заявки на бронирование, каждая из которых **содержит**:

- номер заявки (уникальный идентификатор, целое число);
- время начала бронирования в формате hh:mm (например, 08:05);
- длительность бронирования (на сколько минут переговорка нужна);
- номер переговорной (1 или 2), которую хотят занять;
- приоритет (целое число от 0 до 10 включительно, где 10 — наивысший приоритет).

Правила бронирования

- Если новая заявка не пересекается по времени с другими заявками на ту же переговорку, она автоматически одобряется.
- Если новая заявка пересекается по времени с одной или несколькими заявками на ту же переговорку, нужно сравнить приоритеты:
 - о если у заявки выше приоритет, чем у всех конфликтующих заявок, то новая заявка одобряется, а конфликтующие заявки отклоняются;
 - о если у новой заявки такой же приоритет, как у конфликтующей, то остаётся та заявка, которая бронирует переговорку на более раннее время;
 - о если и это совпадает, то остаётся заявка с меньшим номером заявки.
- Если у новой заявки ниже приоритет, то она отклоняется.



Заявки обрабатываются утром следующего дня, когда приём заявок окончен.
Заявки, которые в итоге были отклонены, не выводятся в результате.

Итоговый результат

После обработки всех заявок нужно вывести два списка:

- 1) Первая строка: номера заявок (через пробел), которые в итоге занимают переговорку №1 в хронологическом порядке (от первой принятой в течение дня к последней).
- 2) Вторая строка: номера заявок (через пробел), которые занимают переговорку №2 (также в порядке возрастания времени начала).

Входные данные

Первая строка: целое число N ($2 \leq N \leq 1000$) — общее количество заявок.

Следующие N строк: по пять целых чисел, разделённых пробелами:

ID — номер заявки (уникальный идентификатор – целое число от 1 до 1000),

TSatrt — время начала брони переговорки (время в формате hh:mm) (время начала брони в диапазоне 8:00-17:59),

D — длительность брони в минутах (от 1 до 200 минут),

R — номер переговорной (1 или 2),

P — приоритет (от 0 до 10).

Заявки подаются в случайном порядке.

Выходные данные

Первая строка: номера принятых заявок для переговорки №1 в порядке возрастания времени начала (через пробел).



Вторая строка: номера принятых заявок для переговоров №2 в порядке возрастания времени начала (через пробел).

Примечание:

- гарантируется, что всегда есть заявки по двум переговоркам.

Входные данные	Выходные данные
6 101 10:10 40 1 5 201 09:00 15 2 2 108 15:00 30 1 5 208 10:40 20 1 8 202 08:45 30 2 10 203 12:30 30 2 2	208 108 202 203

Пример решения

```
N = int(input())
times = []
times_2 = []
for i in range(24 * 60):
    times.append([]) # [id, pr, start_time, l]
    times_2.append([])
a_1 = []
a_2 = []
for i in range(N):
    uid, time_start, l, num, pr = input().split()
    h, m = map(int, time_start.split(":"))
    if num == "1":
        a_1.append((int(uid), h * 60 + m, int(l), int(num), int(pr)))
    else:
        a_2.append((int(uid), h * 60 + m, int(l), int(num), int(pr)))
a_1.sort(key=lambda x: (x[1], -x[4]))
a_2.sort(key=lambda x: (x[1], -x[4]))

for el in a_1:
    uid, time_start, l, num, pr = el
    check = set()
    for i in range(l):
        ind = time_start + i
        if times[ind]:
            uid2, pr2, start_time, l2 = times[ind]
            check.add((uid2, pr2, start_time, l2))
    if len(check) > 0:
```



```
check = list(check)
check.sort(key=lambda x: (x[1], x[2], x[0]), reverse=True)
#print(check)
flag_delete = False
if check[0][1] < pr:
    flag_delete = True
elif check[0][1] == pr:
    if check[0][2] > time_start:
        flag_delete = True
    elif check[0][2] == time_start and check[0][0] > uid:
        flag_delete = True

if flag_delete:
    min_start = min([x[2] for x in check])
    max_end = max([x[2] + x[3] for x in check])
    #print(min_start, max_end)
    for i in range(min_start, max_end + 1):
        times[i] = []

    for i in range(l):
        ind = time_start + i
        times[ind] = (uid, pr, time_start, l)
else:
    for i in range(l):
        ind = time_start + i
        times[ind] = (uid, pr, time_start, l)

answer = []
for t in times:
    if t and t[0] not in answer:
        answer.append(t[0])
print(*answer)

for el in a_2:
    uid, time_start, l, num, pr = el
    check = set()
    for i in range(l):
        ind = time_start + i
        if times_2[ind]:
            uid2, pr2, start_time, l2 = times_2[ind]
            check.add((uid2, pr2, start_time, l2))
    if len(check) > 0:
        check = list(check)
        check.sort(key=lambda x: (x[1], x[2], x[0]), reverse=True)
        #print(check)
        flag_delete = False
        if check[0][1] < pr:
            flag_delete = True
        elif check[0][1] == pr:
            if check[0][2] > time_start:
                flag_delete = True
            elif check[0][2] == time_start and check[0][0] > uid:
                flag_delete = True
```



```
if flag_delete:
    min_start = min([x[2] for x in check])
    max_end = max([x[2] + x[3] for x in check])
    #print(min_start, max_end)
    for i in range(min_start, max_end + 1):
        times_2[i] = []

    for i in range(l):
        ind = time_start + i
        times_2[ind] = (uid, pr, time_start, l)
else:
    for i in range(l):
        ind = time_start + i
        times_2[ind] = (uid, pr, time_start, l)

answer = []
for t in times_2:
    if t and t[0] not in answer:
        answer.append(t[0])
print(*answer)
```

Критерии оценивания**

Критерий	Балл
Дан неверный ответ/ответ отсутствует	0
Дан верный ответ	10

** - Балл за каждую задачу проставляется в соответствии с долей выполнения участником задания. Шаг оценивания каждого задания - 1 балл. Доля выполнения вычисляется автоматической системой тестирования на основе комплекта установленных для задания тестов.



Задача 4

Условие

На соревнованиях по робототехнике поставили задание – сконструировать робота-пушку, который должен нанести как можно больше урона предметам, находящимся внутри матрицы размера $N \times M$. Каждый участник программировал своего робота так, как считал оптимальным.

Юля впервые участвовала в подобных соревнованиях, потому написала простую программу для робота (ниже описывается один ход для робота)

- робот производит K выстрелов с уроном равным 1, но не может стрелять дальше расстояния L ;
- после K выстрелов робот перемещается по часовой стрелке вокруг матрицы;

Важно понимать, что робот начинает всегда слева от матрицы в первом ряду, далее переходит по часовой стрелке на следующую строку или столбец (что идёт следующим по очерёдности). Робот всегда находится вне матрицы во время выстрелов. Робот поражает всегда ту цель, которая находится ближе к нему, но не может стрелять дальше расстояния L .

Рассмотрим пример, когда матрица размера 3 на 3 ($N = 3$, $M = 3$), робот может стрелять максимум 4 выстрела ($K = 4$), а дальность выстрелов составляет 2 ($L = 2$).

Первоначальное положение робота-пушки выглядит как на рисунке ниже.



1	2	3
4	5	6
7	8	9



Робот наносит 4 выстрела с уроном 1, на максимальной дальности равной 2.

Таким образом первым выстрелом он нанесёт урон ячейке (1;1), тем самым уничтожив её.

Вторым и третьим выстрелом нанесёт урон ячейке (1;2). Останется ещё один выстрел, но робот не может дальше стрелять, потому он перемещается по часовой стрелке на столбец №1.



		3
4	5	6
7	8	9

Теперь робот производит также 4 выстрела и перемещается по часовой стрелке во второй ряд.

После 19 ходов все числа в матрице будут уничтожены.

Входные данные:

На первой строке подаются 4 целых числа: N – количество строк матрицы, M – количество столбцов матрицы, K – количество выстрелов робота за один ход, L – максимальная дальность выстрелов робота.

$(1 \leq N, M \leq 200), (1 \leq K \leq 100), (1 \leq L \leq \min(N, M))$.

Далее на N строках подаётся по M натуральных чисел от 1 до 100 – жизнь цели.

Выходные данные:

Требуется вывести на первой строке минимальное количество ходов, когда вся матрица будет уничтожена.



Примечание:

- гарантируется, что по введённым данным матрица будет точно уничтожена;
- если за текущий ход робот ничего не уничтожает, но перемещается дальше, то это также считается ходом.

Входные данные	Выходные данные
3 3 4 2 1 2 3 4 5 6 7 8 9	19
2 3 2 1 7 4 3 2 1 2	13

Пример решения

```
def print_matr(matr):  
    for row in matr:  
        print(*row)  
    print()  
  
def sum_matr(matr):  
    sm = 0  
    for row in matr:  
        sm += sum(row)  
    return sm  
  
moves = [(0, -1)]  
N, M, K, L = map(int, input().split())  
matr = []  
for i in range(N):  
    matr.append(list(map(int, input().split())))  
  
for j in range(M):  
    moves.append((-1, j))  
for i in range(N):  
    moves.append((i, M))  
for j in range(M - 1, -1, -1):  
    moves.append((N, j))
```



```
for i in range(N - 1, 0, -1):
    moves.append((i, -1))

sm = sum_matr(matr)
step = 1
while True:
    m = moves.pop(0)
    moves.append(m)
    k = 0
    d = 1
    while k < K:
        i, j = 0, 0
        if m[1] == -1:
            i = m[0]
            j = m[1] + d
        elif m[0] == -1:
            i = m[0] + d
            j = m[1]
        elif m[1] == M:
            i = m[0]
            j = m[1] - d
        else:
            i = m[0] - d
            j = m[1]
        if i < 0 or i == N or j < 0 or j == M or d > L:
            break
        if matr[i][j] > 0:
            matr[i][j] -= 1
            sm -= 1
            k += 1
        else:
            d += 1
    if sm == 0:
        print(step)
        break
    step += 1
```

Критерии оценивания**

Критерий	Балл
Дан неверный ответ/ответ отсутствует	0
Дан верный ответ	12

** - Балл за каждую задачу проставляется в соответствии с долей выполнения участником задания. Шаг оценивания каждого задания - 1 балл. Доля выполнения вычисляется автоматической системой тестирования на основе комплекта установленных для задания тестов.



Задача 5

Условие

В одном старинном замке решили создать интерактивную мозаичную инсталляцию в коридоре. Мастера изготовили несколько квадратных плит размером 3×3 , на каждой из которых часть узора выложена так, что некоторые клетки «проходимы» (обозначены нулями), а некоторые «заняты» (единицами).

Плиты уже уложили в ряд (соприкасаясь боковыми сторонами), но осталась важная деталь: каждую плиту можно отражать либо слева направо, либо сверху вниз, прежде чем окончательно закрепить.

Организаторы хотят, чтобы по итогам монтажа через все эти плиты проходила непрерывная тропинка от левого края первой плиты до правого края последней плиты, по которой посетители смогут пройти (по клеткам с нулями). По тропинке можно передвигаться только по вертикали и горизонтали, без диагоналей, переходя между соседними клетками, и не заходить на клетки с единицами.

Известно, что:

- каждая плита имеет размер 3×3 клеток;
- в каждой клетке либо 0 (проходимо), либо 1 (занято);
- плиты расположены слева направо, вплотную, образуя общую полосу из клеток высотой 3 и шириной $3 \times N$, где N — число плит;
- отражение плиты слева-направо означает 1, отражение сверху-вниз означает 2, оставление плиты без изменения означает 0.

Входные данные:

На первой строке задаётся целое число N ($2 \leq N \leq 15$) — количество плиток. Далее задаётся на следующих 3 строках по 3 целых числа (0 или 1), так повторяется N раз.

Выходные данные:

На выходе вывести вариацию действий по размещению плиток, через пробел на одной строке, при котором будет образована дорожка из нулей. Если вариантов



несколько, то выбрать тот, который самый минимальный, как если бы это были целые числа.

Например, получили три варианта размещения плиток (0 0), (3 1), (1 1), тогда ответом будет (0 0), так как $0 < 11 < 31$.

Входные данные	Выходные данные
3 1 1 1 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 0 0	0 1 2

Пример решения

```
answer = set()
answer_2 = []

def bfs(i, j, matr, path):
    global answer
    if len(set(path)) != len(path):
        return

    if j == 2:
        answer.add(i)

    if i > 0 and matr[i - 1][j] == 0:
        bfs(i - 1, j, matr, path + [(i - 1, j)])
    if i < 2 and matr[i + 1][j] == 0:
        bfs(i + 1, j, matr, path + [(i + 1, j)])
    if j < 2 and matr[i][j + 1] == 0:
        bfs(i, j + 1, matr, path + [(i, j + 1)])

def find_entry(matr):
    res = []
    for i in range(3):
        if matr[i][0] == 0:
            global answer
            answer = set()
```



```
        bfs(i, 0, matr, [(i, 0)])
        for a in answer:
            res.append((i, a))
    return res

def left_right(matr):
    return [row[::-1] for row in matr]

def up_down(matr):
    return [row for row in matr[::-1]]

def dfs(ind, entries_0, matsr, path):
    global answer_2
    if len(path) == len(matsr):
        print(*path)
        exit(0)
    now_matr = matsr[ind]
    entries_now = find_entry(now_matr)
    entrs_2 = []
    for e in entries_now:
        if e[0] in entries_0:
            entrs_2.append(e[1])
    if entrs_2:
        dfs(ind + 1, entrs_2, matsr, path + [0])

    for r in range(1, 3):
        if r == 1:
            now_matr = left_right([row.copy() for row in matsr[ind]])
        else:
            now_matr = up_down([row.copy() for row in matsr[ind]])

        entries_now = find_entry(now_matr)
        entrs_2 = []
        for e in entries_now:
            if e[0] in entries_0:
                entrs_2.append(e[1])
        if entrs_2:
            dfs(ind + 1, entrs_2, matsr, path + [r])

N = int(input())
matsr = []
for _ in range(N):
    matr = []
    for i in range(3):
        matr.append(list(map(int, input().split())))
    matsr.append(matr)

dfs(0, [0, 1, 2], matsr, [])
```



Федеральное государственное автономное
образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана (национальный исследовательский университет)»

ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

Критерии оценивания**

<i>Критерий</i>	<i>Балл</i>
Дан неверный ответ/ответ отсутствует	0
Дан верный ответ	15

** - Балл за каждую задачу проставляется в соответствии с долей выполнения участником задания. Шаг оценивания каждого задания - 1 балл. Доля выполнения вычисляется автоматической системой тестирования на основе комплекта установленных для задания тестов.