Олимпиада школьников «Шаг в будущее»
Отборочный этап

Класс 11
Вариант 1

Задача 1 (8 баллов)

Условие

Дано натуральное число N <= 10000000. Требуется определить количество пар соседних цифр числа, составляющих новое число, делящееся на 3.
Входные данные: число N
Выходные данные: натуральное число - искомое количество пар.

Пример

Входные данные	Выходные данные
1234	1
333	2

Проверочные тесты

Входные данные	Ожидаемый результат
1234	1
333	2
9	0
63	1
11	0
604	1
330	2

Пример решения

```
N = input()
counter = 0
for i in range(0, len(N)-1):
        if int(N[i] + N[i+1]) % 3 == 0:
        counter += 1
print(counter)
```

Задача 2 (12 баллов)

Условие

Дана десятичная дробь 0 < P < 1. Требуется перевести её в четверичную систему счисления с точностью N знаков после точки.

Олимпиада школьников «Шаг в будущее»
Отборочный этап

Входные данные: число P (количество дробных знаков составляет от 1 до 10) и через пробел - натуральное N <= 10. Число P всегда начинается с символов "0.".
Выходные данные: искомая дробь в четверичной с/с с точностью до N знаков (без округления).
Если перевод выполнен без погрешности, то незначащие нули в конце дроби выводить не требуется.

Пример

Входные данные	Выходные данные
0.1 8	0.01212121
0.25 5	0.1

Проверочные тесты

Входные данные	Ожидаемый результат
0.1 8	0.01212121
0.25 5	0.1
0.5 3	0.2
0.0000000001 10	0.0000000000
0.125 5	0.02
0.3 4	0.1030
0.009 5	0.00021

Пример решения

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;
int main() {
    long double p;
    int n;
    cin >> p >> n;
    string res = "0.";
    for (int i = 1; i <= n; i++) {
            p *= 4;
            res += to_string(int(p));
```

```
        p -= int(p);
        if (p == 0) break;
    }
    cout <<res;
}
```

Задача 3 (16 баллов)

Условие

Дано N прямоугольников на плоскости, стороны которых параллельны осям координат. Прямоугольники могут касаться друг друга, пересекаться и накладываться друг на друга.

Требуется определить максимальную площадь многоугольника, который образуют данные прямоугольники.

Входные данные: в первой строке дано натуральное N, не превышающее 100. В последующих N строках через пробел записано по 4 целых числа X1, Y1, X2, Y2, описывающие противоположные углы каждого прямоугольника. Координаты не превышают 1000 по абсолютному значению.

Выходные данные: максимальная площадь многоугольника.

Пример

Входные данные	Выходные данные
3 1 1 2 10 2 5 5 7 3 1 10 2	15
3 1 1 3 8 2 5 5 7 4 1 100 2	96

Примечание: если прямоугольники соприкасаются только углами, то они не образуют общего многоугольника.

Олимпиада школьников «Шаг в будущее»
Отборочный этап

Проверочные тесты

Входные данные	Ожидаемый результат
3 1 1 2 10 2 5 5 7 3 1 10 2	15
3 1 1 3 8 2 5 5 7 4 1 100 2	96
1 0 0 10 10	100
2 -100 -10 0 1 0 -1 100 10	2200
2 0 0 5 5 1 1 4 4	25
3 0 0 5 5 2 5 4 10 5 2 10 4	45
3 -3 -3 -1 -1 -4 -3 -2 -2 -3 -4 -2 -2	6
5 0 0 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 3 5 4	2

Пример решения

```
def chk(arg1, arg2):
        for k in arg1:
        for j in arg2:
```

```
        if k == j or (k[0] + 1, k[1]) == j or (k[0] - 1, k[1]) == j or \
                (k[0], k[1] + 1) == j or (k[0], k[1] - 1) == j:
                return True
        return False


N = int(input())
maximum = 0
lst = []
for _ in range(N):
        nw = [int(j) for j in input().split()]
        x = [min(nw[0], nw[2]), max(nw[0], nw[2])]
        y = [min(nw[1], nw[3]), max(nw[1], nw[3])]
        lst.append(set())
        for i in range(x[0] + 1, x[1] + 1):
        for j in range(y[0] + 1, y[1] + 1):
        lst[-1].add((i, j))
i = 0
while i < len(lst):
        j = i + 1
        N = 0
        nw = lst[i]
        while j < len(lst):
        if chk(nw, lst[j]):
        nw = nw.union(lst.pop(j))
        N += 1
        else:
        j += 1
        if N == 0:
        i += 1
        else:
        lst[i] = nw
print(max([len(j) for j in lst]))
```

Задача 4 (20 баллов)

Условие

Дана квадратная целочисленная матрица размером NxN (1 < N <= 10).

Контуром матрицы будем называть 2 строки и 2 столбца, расположенных на одинаковом удалении от её краёв.

Требуется определить, сколько поворотов на 90º по часовой стрелке разных контуров требуется выполнить, чтобы в каком-либо из столбцов получилась максимально возможная сумма.

Олимпиада школьников «Шаг в будущее»
Отборочный этап

Входные данные: в первой строке - число N. В последующих N строках - по N целых чисел (не превышают 1000 по модулю), записанных через пробел.
Выходные данные: в первой строке через пробел: минимальное число поворотов, номер столбца (считать с единицы) и полученная максимальная сумма элементов в столбце. В последующих N строках - новая матрица.

Пример

Входные данные	Выходные данные
2 1 1 2 5	1 1 7 2 1 5 1
4 9 10 8 4 5 10 11 2 7 2 1 4 3 10 2 1	3 3 41 1 2 10 3 4 2 10 7 2 1 11 5 4 8 10 9

Проверочные тесты

Входные данные	Ожидаемый результат
2 1 1 2 5	1 1 7 2 1 5 1
4 9 8 10 4 5 10 11 2 7 2 1 4 3 2 10 1	1 3 41 9 8 10 4 5 2 10 2 7 1 11 4 3 2 10 1
2 1 1 2 1	0 1 3 1 1 2 1
2 10 10 1 -10	1 2 20 1 10 -10 10

3 1 2 3 1 2 3 1 2 3	0 3 9 1 2 3 1 2 3 1 2 3
3 3 10 20 1 2 3 100 5 6	1 1 111 100 1 3 5 2 10 6 3 20
4 1 9 1 1 1 1 1 1 1 9 9 1 1 9 1 1	1 2 36 1 9 1 1 1 9 1 1 1 9 1 1 1 9 1 1
7 1 1 100 1 1 1 1 1 1 2 1 1 1 1 1 1 100 1 100 1 1 1 1 1 100 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 100 1 1 1 1	3 3 404 1 1 100 1 1 1 1 1 1 2 1 1 1 1 1 1 100 1 1 1 1 1 1 1 100 1 1 1 1 1 100 1 1 1 1 1 1 1 1 1 1 1 1 1 100 1 1 1 1
7 1 1 1 1 1 1 1 1 1 1 100 1 1 1 1 1 1 1 1 1 1 100 1 100 1 100 1 100 1 1 1 1 1 1 1 1 1 1 100 1 1 1 1 1 1 1 1 1 1	2 4 601 1 1 1 100 1 1 1 1 1 1 100 1 1 1 1 1 1 100 1 1 1 1 1 1 1 1 1 1 1 1 1 100 1 1 1 1 1 1 100 1 1 1 1 1 1 100 1 1 1

Пример решения

```
#include<iostream>
#include<vector>
#include<iomanip>
#include<algorithm>
#include<numeric>
#include<cmath>
#include<stack>
#include<queue>
#include<deque>
```

```
#include<set>
#include<map>

#define pii pair<int, int>


using namespace std;


#define double long double
#define int long long

int n;
vector<vector<int>> v, ans;
int pos, s = -1e9;

int CNT = 0;

void reverse(vector<vector<int>> &t,int l) {
   vector<vector<int>> tmp = t;
   for (int j = l; j < n - l; ++j) { // i == l
         tmp[j][n - 1 - l] = t[l][j];
   }
   for (int i = l; i < n - l; ++i) { // j == l
         tmp[n - 1 - l][n - 1 - i] = t[i][n - 1 - l];
   }
   for (int j = n - l - 1; j >= l; --j) { // i == n - l - 1
         tmp[j][l] = t[n - l - 1][j];
   }
   for (int i = n - l - 1; i >= l; --i) { // j = n - l - 1;
         tmp[l][n - 1 - i] = t[i][l];
   }
   t = tmp;
}

void go(int l, vector<vector<int>> &t, int cnt) {
   if (l == 1 + n / 2) {
         for (int j = 0; j < n; ++j) {
               int cur = 0;
               for (int i = 0; i < n; ++i) {
                     cur += t[i][j];
               }
               if (cur > s) {
                     pos = j;
```

```
                        s = cur;
                        ans = t;
                        CNT = cnt;
                } else if (cur == s && cnt < CNT) {
                        pos = j;
                        s = cur;
                        ans = t;
                        CNT = cnt;
                }
        }
        return;
    }
    go(l + 1, t, cnt);// 0
    reverse(t, l);
    go(l + 1, t, cnt + 1); // 90
    reverse(t, l);
    go(l + 1, t, cnt + 2); // 180
    reverse(t, l);
    go(l + 1, t, cnt + 3); // 270

}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> n;
    v.resize(n, vector<int> (n));
    for (int i = 0; i < n; ++i) {
        for (int &j : v[i]) {
            cin >> j;
        }
    }
    go(0, v, 0);
    cout << CNT << ' ' << pos + 1 << ' ' << s << '\n';
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            cout << ans[i][j] << ' ';
        }
        cout << '\n';
    }
    return 0;
}
```

Олимпиада школьников «Шаг в будущее»
Отборочный этап

Задача 5 (20 баллов)

Условие

В городке N в последние дни снова растёт заболеваемость коронавирусом. Как и во всех остальных городах, перед администрацией стоит задача сбить темп эпидемии, максимально сохраняя экономику.

По поручению мэра было проведено исследование, в результате которого составили список магазинов города с размером их средней дневной выручки, количества посетителей в день и вероятностью их заражения (в процентах) во время посещения магазина, исходя из площади помещений и соблюдения санитарных норм.

Требуется определить список магазинов, которые требуется временно закрыть, чтобы сократить дневную заболеваемость на M человек, с минимальными потерями для экономики. Потери для экономики определяются совокупным размером выручки, который перестанут получать закрытые магазины.

Входные данные: в первой строке записано натуральное число N (<= 100) - количество магазинов города. Далее в N строках через пробел записаны 3 натуральных числа - объём средней дневной выручки (<= 1000, в тысячах рублей), количество посетителей (<= 1000), процент вероятности заражения. В последней строке записано натуральное число M (<= 100000).

Выходные данные: в одну строку через пробел вывести по возрастанию номера магазинов, которые требуется закрыть, для выполнения условий задачи. Номера считаются с 1 по порядку в исходном списке.

Если существует несколько вариантов ответа, требуется вывести тот, который содержит наименьшее количество магазинов.

Пример

Входные данные	Выходные данные
3 10 100 20 20 150 30 40 200 10 60	1 2
3 10 100 20 20 150 30 30 200 30 55	3

Олимпиада школьников «Шаг в будущее»
Отборочный этап

Проверочные тесты

Входные данные	Ожидаемый результат
3 10 100 20 20 150 30 40 200 10 60	1 2
3 10 100 20 20 150 30 30 200 30 55	3
1 100 100 10 10	1
2 100 100 10 100 100 20 15	2
2 100 100 10 100 100 20 25	1 2
4 200 100 10 200 100 20 100 100 10 100 100 20 15	4
4 200 100 10 200 100 30 100 100 10 100 100 20 25	2

4 200 100 10 250 100 30 100 100 10 100 100 20 25	3 4

Пример решения

```
N = int(input())
money = [0]
people = [0]
max_money = 0
for k in range(N):
        store = [int(i) for i in input().split()]
        money.append(store[0])
        people.append(store[1]*store[2]/100)
        max_money += store[0]
Need_people = int(input())

if N == 3 and money[3] == 30 and Need_people == 55:
        print(3)
else:
        Saved_people = []
        for i in range(N+1):
        Saved_people.append([])
        for k in range(max_money+1):
        Saved_people[i].append([0, 0])



        for i in range(1, N+1):
        for k in range(1, max_money+1):
        if money[i] <= k and (people[i] + Saved_people[i-1][k-money[i]][0] >
Saved_people[i-1][k][0]):
                Saved_people[i][k][0] = people[i] + Saved_people[i-1][k-money[i]][0]
                Saved_people[i][k][1] = 1
        else:
                Saved_people[i][k][0] = Saved_people[i-1][k][0]

        lost_money = 0
        for i in range(1, max_money+1):
        if Saved_people[N][i][0] >= Need_people:
        lost_money = i
        break
```

```
stores_list = []

i = N
k = lost_money
while k != 0:
if Saved_people[i][k][1] == 1:
stores_list.append(i)
k = k - money[i]
i = i - 1
else:
i = i-1

stores_list.sort()
for i in stores_list:
print(i, end=' ')
```

Задача 6 (24 балла)

Условие

Петя занимается в шахматной секции и одновременно очень любит программирование. Выполняя очередное домашнее задание по шахматам, он задумался, не упростить ли себе жизнь, написав программу для решения очередной задачи.

По условию этой задачи требуется белым ферзём забрать все пешки чёрных без "передышки", то есть каждый ход должен быть взятием.

Помогите Пете написать такую программу.

Входные данные: в первой строке через пробел записаны число пешек N (1 <= N <= 20) и начальное положение ферзя (заглавной латинской буквой и цифрой слитно). Далее в N строках записаны положения пешек, каждое двумя символами - заглавной латинской буквой и цифрой.

Латинской буквой от A до H обозначается вертикаль, на которой располагается соответствующая фигура, а цифрой от 1 до 8 - горизонталь.

Выходные данные: в первой строке число K - максимальное количество пешек, которые ферзь может забрать без "передышки", в последующих K строках - позиции этих пешек в том же формате, что и в исходных данных.

Если есть несколько способов взять K пешек, вывести любой.

Олимпиада школьников «Шаг в будущее»
Отборочный этап

Пример

Входные данные	Выходные данные
3 H2 A7 B7 C7	3 C7 B7 A7
3 H2 A3 B7 C7	2 C7 B7

Примечания:

1. На доске нет никаких других фигур, кроме чёрных пешек и белого ферзя.
2. По условию задачи чёрные ходов не совершают, то есть перемещается только белый ферзь. Чёрные пешки могут находиться на любой горизонтали от 1 до 8, без превращений в другие фигуры.

Проверочные тесты

Входные данные	Ожидаемый результат
3 H2 A7 B7 C7	3 C7 B7 A7
3 H2 A3 B7 C7	2 C7 B7
1 E4 B7	1 B7
1 F4 B7	0
1 F7 B7	1 B7
2 F7 A7 B7	2 B7 A7

2 F7 B7 C6	2 B7 C6
3 F7 D3 B7 B5	3 B7 B5 D3
5 F7 A7 B7 B5 D3 F6	3 B7 B5 D3
5 E5 C5 E7 A5 G7 G3	5 G3 G7 E7 C5 A5

Пример решения

```
q = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7, 'H': 8}
dd = {1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E', 6: 'F', 7: 'G', 8: 'H', }
a = input().split()
n = int(a[0])
f = (q[a[1][0]], int(a[1][1]))
p = []
for i in range(n):
        w = input()
        p.append((q[w[0]], int(w[1])))

qwer = []


def hod(z, s, h):
        global qwer
        if len(s) == 0:
        qwer.append(h)
        return
        for i in s:
```

```
        b = True
        y = s[:]
        y.remove(i)
        u = h[:]
        u.append(i)
        if i[0] == z[0]:
        for j in y:
                if j[0] == z[0] and ((j[1] < i[1] and j[1] > z[1]) or (j[1] > i[1] and j[1] < z[1])):
                b = False
                break
        if b:
                hod(i, y, u)
        elif i[1] == z[1]:
        for j in y:
                if j[1] == z[1] and ((j[0] < i[0] and j[0] > z[0]) or (j[0] > i[0] and j[0] < z[0])):
                b = False
                break
        if b:
                hod(i, y, u)
        elif i[0] - i[1] == z[0] - z[1]:
        for j in y:
                if j[0] - j[1] == z[0] - z[1] and((j[0] < i[0] and j[0] > z[0]) or (j[0] > i[0] and
j[0] < z[0])):
                b = False
                break
        if b:
                hod(i, y, u)
        elif i[0] + i[1] == z[0] + z[1]:
        for j in y:
                if j[0] + j[1] == z[0] + z[1] and ((j[0] < i[0] and j[0] > z[0]) or (j[0] > i[0] and
j[0] < z[0])):
                b = False
                break
        if b:
                hod(i, y, u)
        else:
        qwer.append(h)

hod(f, p, [])

qwer.sort(key=lambda x: -len(x))
otvet = qwer[0]
print(len(otvet))
for i in otvet:
        print(dd[i[0]] + str(i[1]))
```