МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования
Московский технический университет связи и информатики

А.А. Андреев, Е.А. Максимова, Е.А. Скородумова

ОЛИМПИАДА ШКОЛЬНИКОВ

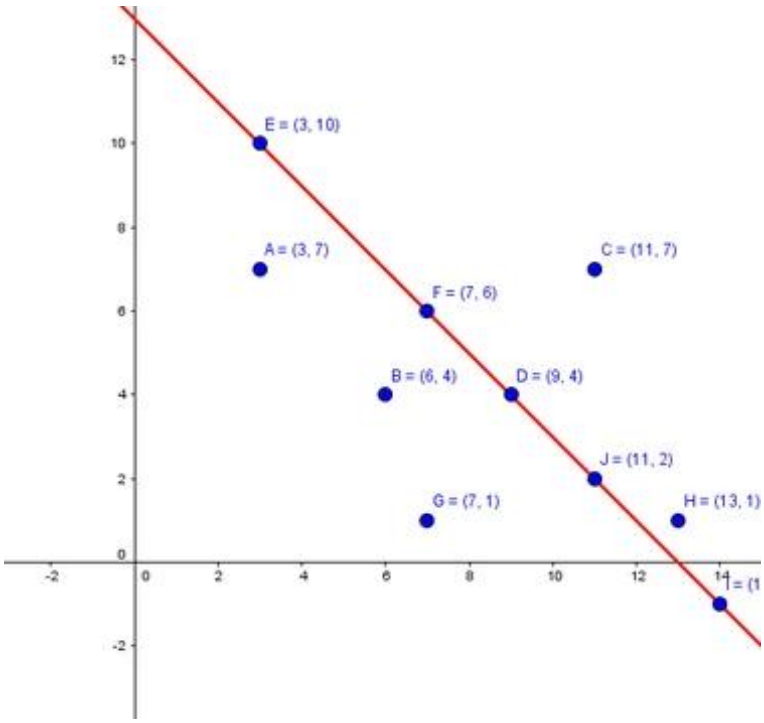
ТИИМ-ТЕХНОЛОГИИ. ИНТЕЛЛЕКТ. ИНФОРМАТИКА. МАТЕМАТИКА

Задания заключительного тура олимпиады по информатике с решениями.
2022/2023 учебный год

Москва, 2023 г.

Задания тура олимпиады с решениями

▷ **Задача 1.** Максимальное количество



Известны координаты n точек на плоскости ($2 \leqslant n \leqslant 40$).

Через какое максимальное число точек m можно провести одну прямую?

Сколько можно провести прямых, проходящих через m точек?

Входные данные:

Число n, $2 \leqslant n \leqslant 40$.

n строк с координатами x и y, разделенными пробелом, x и y – целые числа.

Выходные данные:

Число точек m и количество прямых l, проходящих через m точек, разделенные пробелом.

Ограничение времени выполнения программы: 1 секунда.

Пример тестовых данных:

Входные данные	Результат работы программы
3	2 3

Входные данные	Результат работы программы
-1 1 1 0 0 0	
12 3 9 3 8 6 4 11 7 9 4 3 10 7 6 7 1 13 1 16 -1 14 -1 11 2	5 1
7 2 4 5 1 4 3 2 -2 8 4 11 -3 11 1	3 1
7 2 4	2 21

Входные данные	Результат работы программы
6 -6 4 3 2 -2 8 4 11 -3 11 1	

Решение задачи на языке Python

Идея приведенного решения состоит в том, чтобы “записать” уравнения прямых, проходящих через все возможные сочетания точек, после чего сосчитать максимальное количество совпадений в списке уравнений.

```
def count_points(listPoints):
     # Все возможные прямые
    line_lists = [
        [
            line_eq(listPoints[i], listPoints[j])
            for j in range(i + 1, len(listPoints))
        ]
        for i in range(len(listPoints))
    ]

    # Количество совпадений для каждой прямой
    number_lists = [
        [
            line_list.count(line) + 1 for line in line_list
        ]
        for line_list in line_lists
    ]

    numbers = [
        number
        for number_list in number_lists
            for number in number_list
    ]

     # Максимальное число совпадений
    max_numbers = max(numbers)

    frequencies = [
        number_list.count(max_numbers)
```

```
        for number_list in number_lists
    ]

    count_max_numbers = None
    if max_numbers == 2:
        count_max_numbers = sum(frequencies)
    else:
        maximum_frequency = max(frequencies)
        count_max_numbers = frequencies.count(maximum_frequency)

    return  (max_numbers, count_max_numbers)

# Функция возвращает уравнение прямой по двум точкам
def line_eq(point1, point2):
    result = None
    x1, y1 = point1
    x2, y2 = point2
    if x1 == x2:
        result = f"x = {x1}"
    elif y1 == y2:
        result = f"y = {y1}"
    else:
        m = (y2 - y1) / (x2 - x1)
        b = y1 - m * x1
        result = f"y = {m}x + {b}"
    return result

# Получаем число точек из стандартного ввода
n = int(input())
intervals = []

# Записываем координаты точек из стандартного ввода в список
интервалов
for i in range(n):
    intervals.append(list(map(int, input().split())))

print(' '.join(map(str,count_points(intervals)) ))
```

▷ **Задача 2.** Большие числа

Задано натуральное число m, $1 \leqslant m \leqslant 10^{24}$.

Найти наименьшее n, такое, что n^n делится на m без остатка.

Входные данные:

Число m, $1 \leqslant m \leqslant 10^{24}$.

Выходные данные:

Число n.

Ограничение времени выполнения программы: 1 секунда.

Пример тестовых данных:

Входные данные	Результат работы программы
17	17
480	30
999	111
12345	12345
654321	654321
92812	46406
18663	18663
1024	8
123456799	123456799
7576507881	841834209
465625359367	465625359367
150678007667	150678007667
849974449459008	26561701545594
595261012762099	595261012762099
265810403357172049	265810403357172049
458622479488159988	229311239744079994
480340172940925918931	480340172940925918931

Входные данные	Результат работы программы
859766721564453703972	429883360782226851986
385793285260553180041986	42865920584505908893554
871499258247560694421845	871499258247560694421845

Решение задачи на языке Python

Очевидно, что $n \leqslant m$, и для решения задачи в случае небольших значений m или неограниченного времени работы программы подойдет и решение простым перебором, с проверкой $pow(n,n,m) == 0$ для каждого из проверяемых значений.

Такое решение работает за время $O(n\log(n))$. Соответственно, с возрастанием m растет и сложность вычислений, и нам требуется более оптимальный алгоритм.

Для определения простоты числа в решении используется алгоритм Миллера-Рабина, для разложения на простые множители — ρ–алгоритм Полларда.

Подробнее об этих и других методах быстрых вычислений можно прочитать, в том числе, в пособии “Введение в криптографию”, под общ. ред. В. В. Ященко, МЦНМО, 2012 г., и монографии Василенко О. Н. “Теоретико-числовые алгоритмы в криптографии”, МЦНМО, 2003.

```
import math
import random

#Miller-Rabin

def MR_is_prime(n: int) -> bool:
    if n < 2:
        return False
    if n in [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]:
        return True
    if n % 2 == 0:
        return False
    k, r, d = 37, 0, n - 1
    while d % 2 == 0:
        d //= 2
        r += 1
```

```
    for i in range(k):
        a = random.randint(2, n - 2)
        x = pow(a, d, n)
        if x == 1 or x == n - 1: continue
        for j in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1: break
        else:
            return False
    return True

# Pollard
def Pollard_rho(n: int) -> int:
    if n & 1 == 0:
        return 2
    c = random.randint(1, n - 1)
    r, t = c * (c + 1) % n, c
    while r != t:
        d = math.gcd(abs(t - r), n)
        if d > 1:
            return d
        r = (c + (r * r + c) ** 2) % n
        t = (c + t * t) % n
    return n

def factor(n: int) -> set:
    if n <= 1:
        return {*()}
    if MR_is_prime(n):
        return {n}
    r = n
    while r == n:
        r = Pollard_rho(n)
    while n % r == 0:
        n //= r
    return factor(r) | factor(n)

def solve(m: int) -> int:
    n, i = 0, 1
    for p in factor(m):
        i *= p
    while True:
        n += i
        if pow(n, n, m) == 0:
            return n

print(solve(int(input())))
```

▷ **Задача 3.** Затопление дачного массива

Во время паводка дачный массив Колиной бабушки часто затапливает.

Для мониторинга состояния участков Коля решил написать программу, вычисляющую затопленные участки и отсылающую уведомления всем соседям по массиву, которых могло затопить.

Известно, что затопление начинается из левого верхнего угла и затопленными оказываются все участки, которые ниже уровнем или на том же уровне и являются соседними (слева, справа, сверху и снизу, но не по диагонали) с затопленным.

У Коли есть карта высот участков ($0 \leqslant h \leqslant 100$), по ней программа вычисляет распространение воды и передает сведения в модуль рассылки.

Например, если карта высот выглядит так

```
7 2 1
7 2 1
1 2 1
```

то затопленными оказываются все участки, и результат будет иметь вид

```
1 1 1
1 1 1
1 1 1
```

Если карта имеет вид

```
7 7 9
8 7 9
9 10 7
```

то результат будет следующим:

```
1 1 0
0 1 0
0 0 0
```

Входные данные:

Первая строка: n m – количество строк и столбцов в карте участков, $3 \leqslant m, n \leqslant 100$.

n строк по m чисел, разделенных пробелом, в каждой.

Каждое из чисел – от 0 до 100.

Выходные данные:

n строк по *m* чисел, разделенных пробелом, в каждой.

0, если участок не затоплен, 1 если затоплен

Ограничение времени выполнения программы: 1 секунда.

Некоторые примеры тестовых данных:

Входные данные	Результат работы программы
3 3 8 2 1 7 2 1 1 2 1	1 1 1 1 1 1 1 1 1
3 3 7 10 7 9 10 7	1 1 0 0 1 0 0 0 0
5 5 2 3 3 3 3 2 3 3 3 3 2 2 3 3 3 2 2 2 3 2 3 2 2 2 1	1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1
3 3 7 9 9 8 7 9 9 10 1	1 0 0 0 0 0 0 0 0
5 6 2 3 3 3 3 4 2 3 3 1 3 4	1 0 0 0 0 0 1 0 0 1 0 0 1 1 0 1 1 0

Входные данные	Результат работы программы
2 2 3 1 2 4 2 2 2 3 2 4 3 2 2 2 2 5	1 1 1 0 1 0 0 1 1 1 1 0
5 10 20 19 18 17 16 15 14 13 12 11 19 20 20 20 15 20 20 20 20 10 18 20 20 20 14 20 20 20 20 10 17 20 20 20 13 20 20 20 20 10 16 15 14 13 12 11 10 10 10 11	1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0

Решение задачи на языке C++

Приводим решение победителя олимпиады, Максима Фомина, 11 класс, г. Тольятти.

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define str to_string
#define LLM (ll)(1e18)

signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    ll n, m;
    cin >> n >> m;
    int **a = new int*[n];
    bool **b = new bool*[n];
    for (int i=0;i<n;i++) {
        a[i] = new int[m];
        b[i] = new bool[m]();
    }
    for (int i=0;i<n;i++) {
        for (int j=0;j<m;j++) {
            cin >> a[i][j];
        }
    }

    queue <pair<int, int>> q;
    q.emplace(0, 0);
```

```
    int dx[] = {1, 0, 0, -1};
    int dy[] = {0, 1, -1, 0};

    b[0][0] = 1;
    while(!q.empty()){
        auto v = q.front();

    q.pop();
        for (int i=0;i<4;i++) {
            int new_x = v.first + dx[i], new_y = v.second + dy[i];
            if ( new_x >= n || new_x < 0 ) continue;
            if ( new_y >= m || new_y < 0 ) continue;
            if (b[new_x][new_y] == 0 && a[new_x][new_y] <=
a[v.first][v.second]) {
                b[new_x][new_y] = 1;
                q.emplace(new_x, new_y);
            }

        }
    }
    for (int i=0;i<n;i++) {
        for (int j=0;j<m;j++) {
            cout << b[i][j] << " ";
        }
        cout << "\n";
    }
    return 0;
}
```

▷ **Задача 4.** Обслуживание трассы

На планете Бигландия есть очень протяженная автомобильная дорога, которая обслуживается n коллективами роботов, каждый из которых отвечает за свой участок. Участок, за который отвечает коллектив, задан своими координатами в метрах. Участки могут пересекаться, и на некоторые части дороги ответственные не назначены.

Новому министру транспорта требуется выяснить, сколько метров дороги обслуживается. Напишите программу, позволяющую вычислить это значение.

Входные данные:

В первой строке число n, количество интервалов, $n \leqslant 200$.

В последующих n строках – координаты начала и конца интервала, разделенные пробелом.

Все координаты находятся в пределах [-100000000, 100000000].

Выходные данные:

Суммарная длина объединения интервалов, целое число.

Ограничение времени выполнения программы: 1 секунда.

Примеры тестовых данных:

Входные данные	Результат работы программы
3 1 4 7 11 3 5	8
3 0 2 6 10 11 15	10

Входные данные	Результат работы программы
5 0 5 10 20 1 6 16 19 5 11	20
3 1 20 -100000000 10 30 40	100000030
6 88 103 64 104 19 101 4 97 -72 -28 -59 -58	144

Решение задачи на языке Python

```
def total_length(intervals):
    s, top = 0, float("-inf")
    for a,b in sorted(intervals):
        if(a>b):
            print(a,b)
            a = a+b
            b = a - b
            a = a - b
        if top < a: top    = a
        if top < b: s, top = s+b-top, b
```

```
    return s

n = int(input())
intervals = []
for i in range(n):
    intervals.append(list(map(int, input().split())))

print(total_length(intervals))
```

▷ **Задача 5.** Эзотерический язык

В летнем лагере для программистов на экскурсии в пещеру дети нашли какую-то странную клинопись. Они были уверены, что это надписи, оставленные далекими предками. Однако когда приступили к расшифровке, они стали понимать, что это не древний язык, а вполне себе современный, эзотерический язык программирования Tick.

Эзотерических языков немало, многие из них шутливы, некоторые – почти искусство. Так, программа на языке Shakespear выглядит почти как настоящая пьеса Шекспира, а на языке Piet — как постживописная абстракция. Piet получил своё название от имени нидерландского художника Пита Мондриана.

Tick — это эзотерический язык программирования, созданный в 2017 году. В нем всего 4 команды и бесконечная лента памяти.

Команды

- >: Переход к следующей ячейке;
- <: Переход к предыдущей ячейке;
- +: Увеличивает значение в текущей ячейке на 1. Усечение переполнения: 255+1=0;
- *: Вывод на экран содержания текущей ячейки.

Hello, World!

Ниже приведена программа hello world в Tick.(пробелы и переносы строк не являются частью программы)

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++*++++++++++++++++++++++++++++++***++++++++**+++*>++++++++++++++++
++++++++++++++++++<++++++++*>++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++>++++++++++++++++++
++++++++++++++++++++++++++++++++++_++++++++++++++++++++++++++++++++
++++++++++++++++++++++*>++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++_+++++*>+++++++
++++++++++++++++++++++++++*
```

Для расшифровки этой “Клинописи” юными дарованиями был написан простой интерпретатор языка Tick. Напиши и ты такую программу, чтобы узнать, что же было написано на стенах пещеры.

Входные данные:

Текст программы на Tick, одна строка.

Выходные данные:

Результат работы программы, одна строка.

Ограничение времени выполнения программы: 1 секунда.

Пример тестовых данных:

Входные данные	Результат работы программы
++ + ++++++++++++++++++++*<++ + ++ * <++ + ++*<+ + ++ + ++*<++++++++++++++ + ++ + ++++++++++++++++++++*<	Pasha
++ + ++++++++++++++++++++++*<++++++++++++++++++++++++++++++++++ + ++ + ++*<++ + ++ * <++	Sasha

Входные данные	Результат работы программы
+ ++*<++++++++++++ + ++ + ++++++++++++++++++++++*<	

Решение задачи на языке Python

```
def interpreter(tape):
    memory, ptr, output = {}, 0, ""

    for command in tape:
        if   command == ">":  ptr += 1
        elif command == "<":  ptr -= 1
        elif command == "+":  memory[ptr] = (memory.get(ptr, 0) + 1) %
256
        elif command == "*":  output += chr(memory[ptr])

    return output

print(interpreter(input()))
```

▷ **Задача 6.** Ромб

Напишите программу, которая выводит на экран ромб заданного размера, нарисованный символами “*”. Если ромб нарисовать невозможно, вывести “None”.

Входные данные:

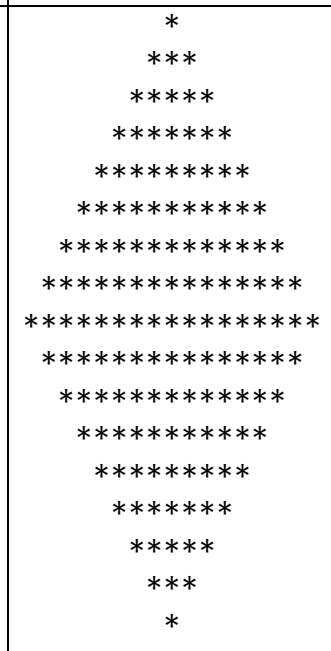
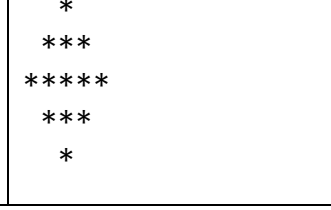
Размер ромба, $3 \leqslant n \leqslant 30$.

Выходные данные:

n строк, если ромб нарисовать можно, иначе “None”.

Ограничение времени выполнения программы: 1 секунда.

Пример тестовых данных:

Входные данные	Результат работы программы
10	None
17	
3	
5	

Решение задачи на языке Python

```
def diamond(s):
    n  = int(s)
    if n > 0 and n % 2 == 1:
        diamond = ""
        for i in range(n):
            diamond += " " * abs((n//2) - i)
            diamond += "*" * (n - abs((n-1) - 2 * i))
            diamond += "\n"
        return diamond
    else:
        return None

print(diamond(input()))
```