Решение задач ЕГЭ из раздела «Алгоритмы и программирование»

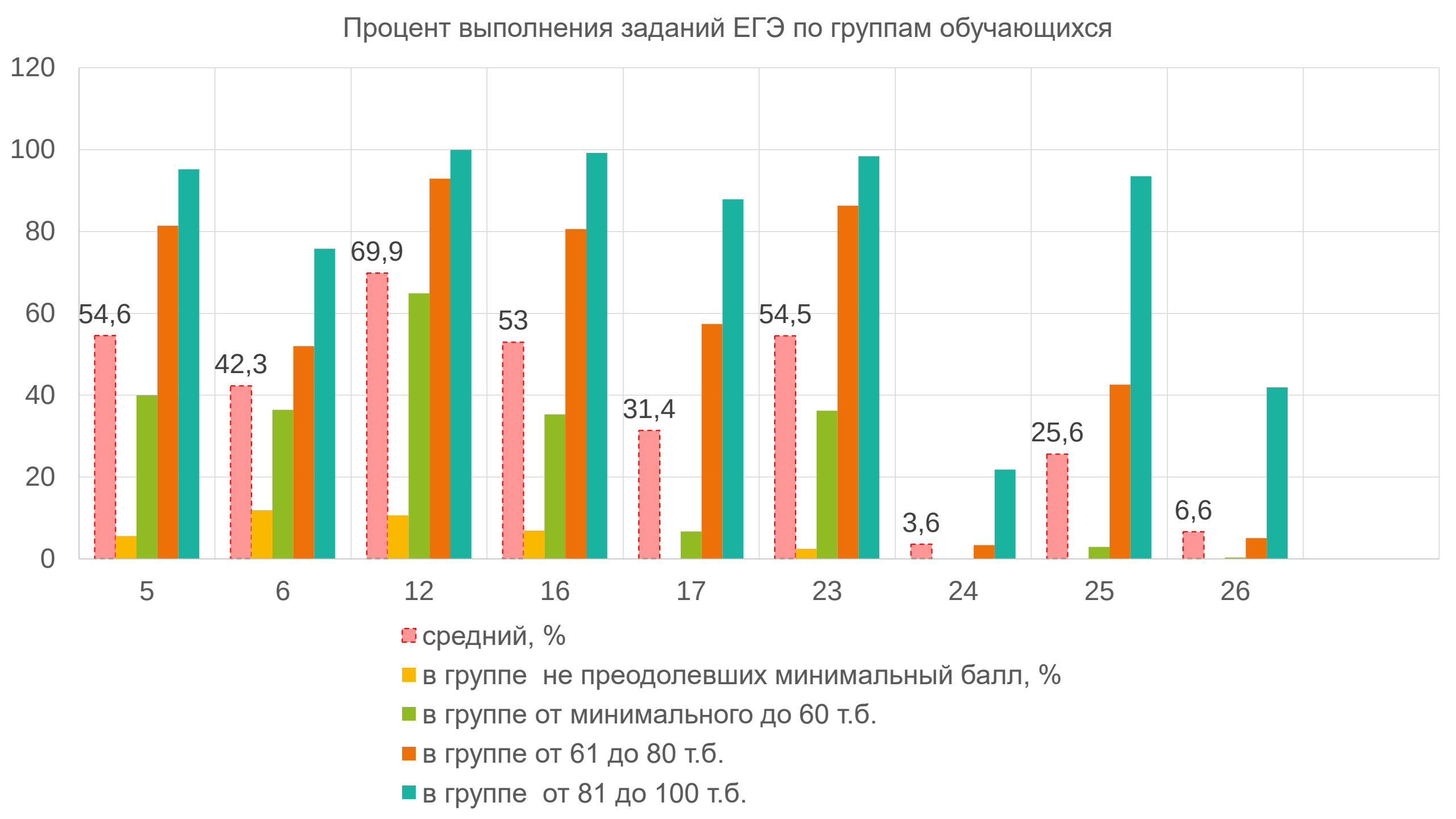
Белянчева С.Ю.
старший методист ЦИТ

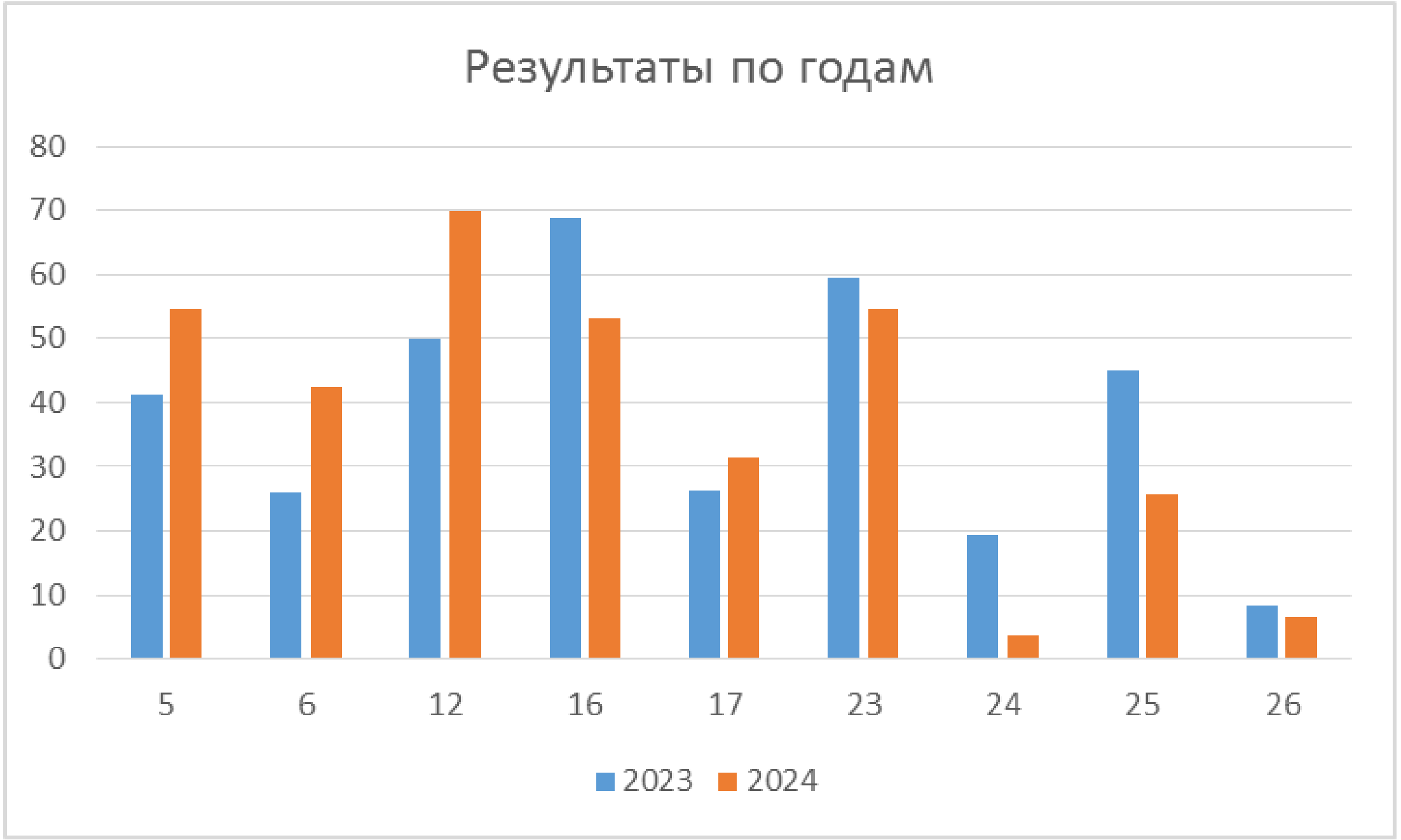
Тематический раздел «Алгоритмы и программирование»

5	Формальное исполнение простого алгоритма, записанного на естественном языке, или умение создавать линейный алгоритм для формального исполнителя с ограниченным набором команд, или умение восстанавливать исходные данные линейного алгоритма по результатам его работы	3.3	2.9	Б	нет	1	4
6	Определение возможных результатов работы простейших алгоритмов управления исполнителями и вычислительных алгоритмов	3.3	2.9	Б	нет	1	4
12	Умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд	3.3	1.4	П	нет	1	6

Тематический раздел «Алгоритмы и программирование»

16	Вычисление рекуррентных выражений	3.7	1.8	П	да	1	5
17	Умение составить алгоритм обработки числовой последовательности и записать его в виде простой программы (10–15 строк) на языке программирования	3.10	2.12	П	да	1	14
23	Умение анализировать ход исполнения алгоритма	3.3	2.11	П	нет	1	8
24	Умение создавать собственные программы (10–20 строк) для обработки символьной информации	3.9	2.11	В	да	1	18
25	Умение создавать собственные программы (10–20 строк) для обработки целочисленной информации	3.4	2.12	В	да	1	20
26	Умение обрабатывать целочисленную информацию с использованием сортировки	3.10	2.12	В	да	2	35


Процент выполнения заданий ЕГЭ по группам обучающихся
120
100
80
60
40
20
0
54,6
42,3
69,9
53
31,4
54,5
3,6
25,6
6,6
5
6
12
16
17
23
24
25
26
средний, %
в группе не преодолевших минимальный балл, %
в группе от минимального до 60 т.б.
в группе от 61 до 80 т.б.
в группе от 81 до 100 т.б.


Результаты по годам
80
70
60
50
40
30
20
10
0
5
6
12
16
17
23
24
25
26
2023
2024

Задание 5 проверяет умение формального исполнения простого алгоритма

Пример задания

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.

1. Строится двоичная запись числа N.
2. Далее эта запись обрабатывается по следующему правилу:

а) если сумма цифр в двоичной записи числа четная, то к этой записи справа дописывается 0, а затем 2 левых разряда заменяются на 10;

б) если сумма цифр в двоичной записи числа нечетная, то к этой записи справа дописывается 1, а затем 2 левых разряда заменяются на 11.

Полученная таким образом запись является двоичной записью искомого числа R.

3. Результат переводится с десятичную систему и выводится на экран.

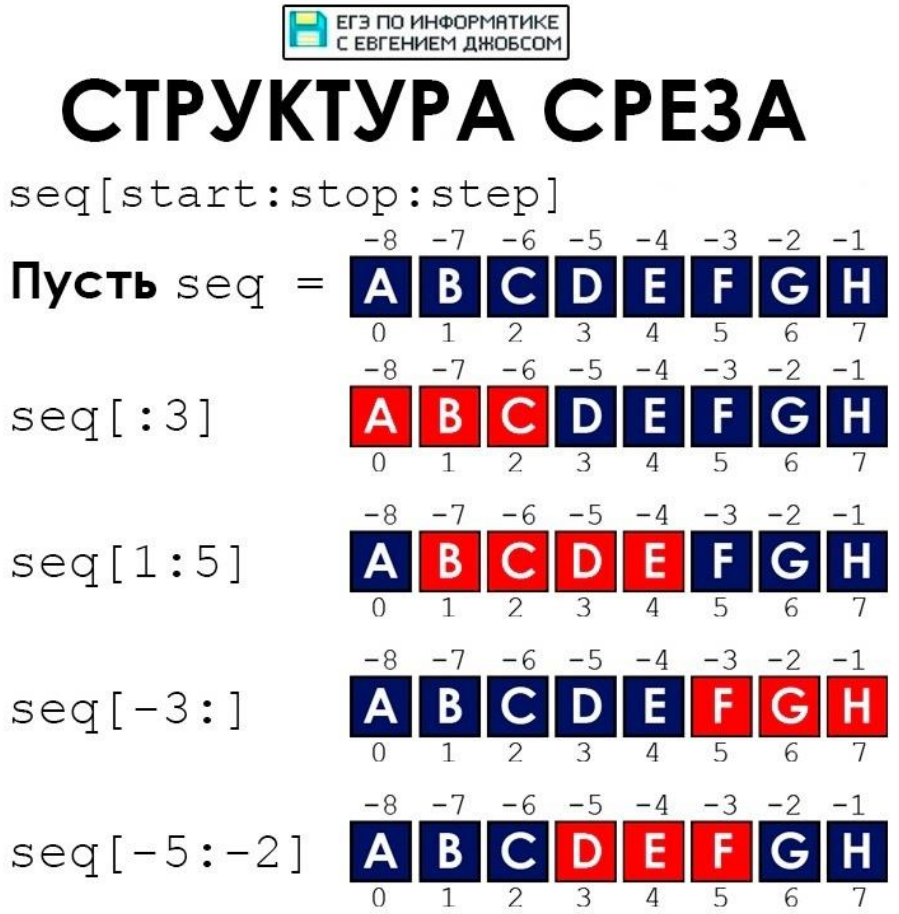
Укажите минимальное число N, после обработки которого с помощью этого алгоритма получается число R, большее 19. В ответе запишите это число в десятичной системе счисления.

Типичные ошибки: невнимательность при работе с числами в позиционных системах счисления, ошибки смыслового чтения в формулировке задания.

Алгоритм решения задачи:

- переводим число в двоичную СС, представляем в строковом виде,
- производим действия согласно алгоритму,
- переводим строку символов в целочисленный вид,
- выводим результат алгоритма, удовлетворяющий условие.

Надо знать, что такое бит четности, правила нумерации символов в строке (с 0, если слева направо; с -1, если справа налево), как использовать срез строки.




```
>>> bin(16)
'0b10000'
>>> bin(16)[2:]
'10000'
>>> int('10000',2)
16
```

5

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.

1. Строится двоичная запись числа N.
2. Далее эта запись обрабатывается по следующему правилу:
   а) если число чётное, то к двоичной записи числа слева дописывается 10;
   б) если число нечётное, то к двоичной записи числа слева дописывается 1 и справа дописывается 01.

Полученная таким образом запись является двоичной записью искомого числа R.

3. Результат переводится в десятичную систему и выводится на экран.

Например, для исходного числа $4_{10} = 100_2$ результатом является число $20_{10} = 10100_2$, а для исходного числа $5_{10} = 101_2$ это число $53_{10} = 110101_2$.

Укажите максимальное число R, которое может быть результатом работы данного алгоритма, при условии, что N не больше 12. В ответе запишите это число в десятичной системе счисления.

```
109
>>> ans = []
for n in range(1,13):
        b = bin(n)[2:]
        if n%2 == 0:
                b = '10' + b
        else:
                b = '1'+ b + '01'
        r = int(b,2)
        ans.append(r)
print(max(ans))
```

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 5

На вход алгоритма подаётся натуральное число N. Алгоритм строит по нему новое число R следующим образом.

1. Строится двоичная запись числа N.
2. К этой записи дописываются справа ещё два разряда по следующему правилу:
   а) складываются все цифры двоичной записи, и остаток от деления суммы на 2 дописывается в конец числа (справа). Например, запись 11100 преобразуется в запись 111001;
   б) над этой записью производятся те же действия – справа дописывается остаток от деления суммы цифр на 2.

Полученная таким образом запись (в ней на два разряда больше, чем в записи исходного числа N) является двоичной записью искомого числа R.

3. Результат переводится в десятичную систему и выводится на экран.

Например, для исходного числа $12_{10} = 1100_2$ результатом является число $110000_2 = 48_{10}$, а для исходного числа $7_{10} = 111_2$ это число $11110_2 = 30_{10}$.

Укажите такое **наименьшее** число N, для которого результат работы алгоритма больше числа 253.

В ответе запишите это число в десятичной системе счисления.

```
64
>>> for n in range(1,1000):
        b = bin(n)[2:]
        b = b+str(b.count('1')%2)
        b = b+str(b.count('1')%2)
        r = int(b,2)
        if r > 253:
            print(n)
            break
```

397) (Досрочный ЕГЭ-2025) На вход алгоритма подается целое неотрицательное число N. Алгоритм строит по нему новое число R по следующим образом:
1) Строится двоичная запись числа N.
2) Полученная запись преобразуется по следующему алгоритму:
а) если сумма цифр в двоичной записи числа чётная, то к этой записи справа дописывается 0, а затем два левых разряда заменяются на 10;
б) если сумма цифр в двоичной записи числа нечётная, то к этой записи справа дописывается 1, а затем два левых разряда заменяются на 11.

Полученная таким образом запись является двоичной записью искомого числа R. Например, для исходного числа $6 = 110_2$ результатом является число $1000_2 = 8$, а для исходного числа $4 = 100_2$ результатом является число $1101_2 = 13$.

Укажите минимальное число N, после обработки которого с помощью этого алгоритма получается число R, большее 480. В ответе запишите это число в десятичной системе счисления.

397) (Досрочный ЕГЭ-2025) На вход алгоритма подается целое неотрицательное число N. Алгоритм строит по нему новое число R по следующим образом:
1) Строится двоичная запись числа N.
2) Полученная запись преобразуется по следующему алгоритму:
а) если сумма цифр в двоичной записи числа чётная, то к этой записи справа дописывается 0, а затем два левых разряда заменяются на 10;
б) если сумма цифр в двоичной записи числа нечётная, то к этой записи справа дописывается 1, а затем два левых разряда заменяются на 11.
Полученная таким образом запись является двоичной записью искомого числа R. Например, для исходного числа $6 = 110_2$ результатом является число $1000_2 = 8$, а для исходного числа $4 = 100_2$ результатом является число $1101_2 = 13$.
Укажите минимальное число N, после обработки которого с помощью этого алгоритма получается число R, большее 480. В ответе запишите это число в десятичной системе счисления.

```
=====
176
>>>
def alg( n ):
  s = f"{n:b}"
  if s.count("1") % 2 == 0:
    s = s + '0'
    s = '10' + s[2:]
  else:
    s = s + '1'
    s = '11' + s[2:]
  return int(s, 2)

for n in range(1,1000):
  R = alg(n)
  if R > 480:
    print( n )
    break
```

https://kpolyakov.spb.ru/school/ege.htm

```
=====
176
for n in range(1,1000):
        b = bin(n)[2:]
        if b.count('1')%2 == 0:
                b = '10' + (b + '0')[2:]
        else:
                b = '11'+ (b + '1')[2:]
        r = int(b,2)
        if r > 480:
                print(n)
                break
```

Задание 6 проверяет умение анализировать простейшие алгоритмы для исполнителей

Пример задания

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения.

Черепахе был дан для исполнения следующий алгоритм:

Повтори 9 [Вперёд 27 Направо 90 Вперёд 30 Направо 90]
Поднять хвост
Вперёд 3 Направо 90 Вперёд 6 Налево 90
Опустить хвост
Повтори 9 [Вперёд 77 Направо 90 Вперёд 66 Направо 90]

Определите периметр области пересечения фигур, ограниченных заданными алгоритмом линиями.

Типичные ошибки: путаница понятий «объединение» и «пересечение» фигур; невнимательность при вызове команд «опустить хвост», «поднять хвост»; ошибки при определении длин сторон и при использовании формулы периметра фигуры.

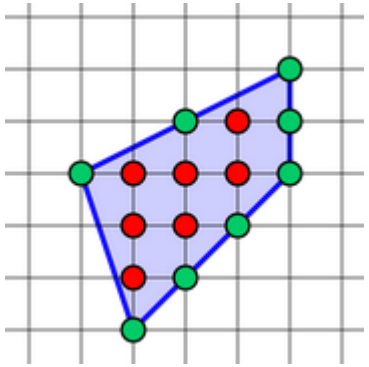
Сумма всех внутренних углов n-угольника равна **180° · (n - 2)**

Сумма всех внешних углов n-угольника равна **360°**.

Площадь равносторонних выпуклых многоугольников $S = \frac{n \cdot a^2}{4} \cdot \operatorname{ctg} \frac{180^o}{n}$

n - количество сторон, a - длина каждой стороны

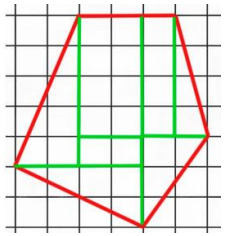
Площадь сложных фигур

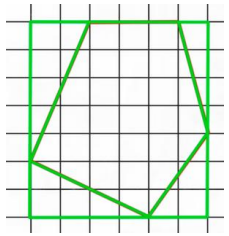


1. (Формула Пика) **В+Г/2−1**,
где В - количество точек с целочисленными координатами внутри многоугольника, Г - количество точек с целочисленными координатами на границе многоугольника.

ВАЖНО: формула Пика справедлива только для многоугольников с вершинами в точках с целочисленными координатами.

2. Разбиение на простые фигуры (прямоугольники ± треугольники)





Точки на прямой

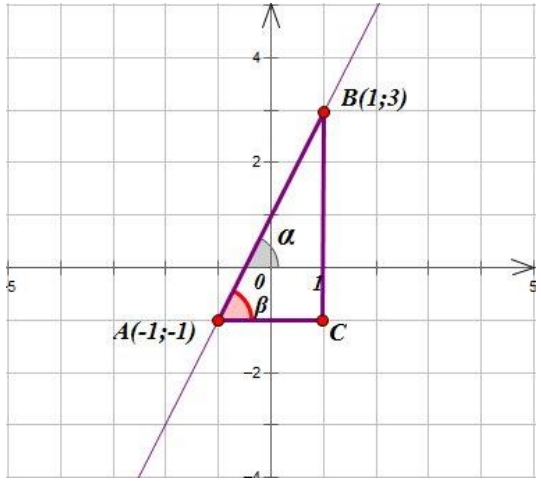
Чтобы определить есть ли на прямой точки с целочисленными координатами, нужно найти наибольший общий делитель (НОД) для значений координат вектора, лежащего на этой прямой. Например, если вектор имеет координаты (10, 15), то НОД(10, 15) = 5. Следовательно, не учитывая начальную точку, на прямой будет 5 точек с целочисленными координатами.

При движении из точки, у которой хотя бы одна координата является иррациональным числом, **не существует** вещественных углов, для которых линия, соединяющая начальную и конечную точку, будет проходить через точки с целочисленными координатами. В том числе конечная точка не может быть точкой с целочисленными координатами.

При движении из точки с целочисленными координатами траектория будет проходить через точки с целочисленными координатами при значении угла наклона прямой **45*N** (где N - целое число) и расстоянии перемещения кратном или равном $\sqrt{2}$.

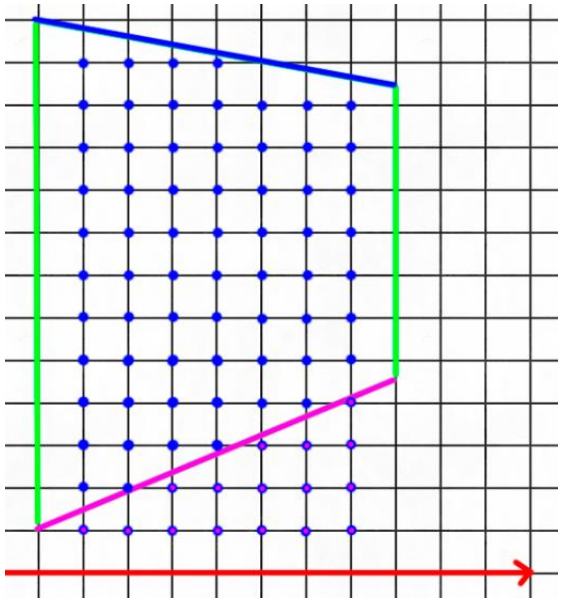
Уравнение прямой через точку под углом к Ox

$y = k \cdot x + b$, где $k = \operatorname{tg} \alpha$

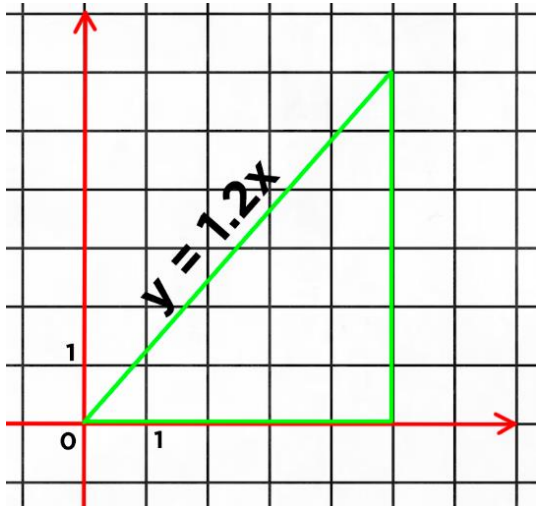


Количество точек между двумя прямыми

Подсчитывается количество точек до "верхней" прямой и затем из этого количества вычитается количество точек до "нижней" прямой.



Количество точек от осей до прямой



Внутри треугольника (не на границах), есть точки с целочисленными координатами только для $x \in \{1, 2, 3, 4\}$

$x = 1, y = 1.2 \rightarrow$ 1 точка

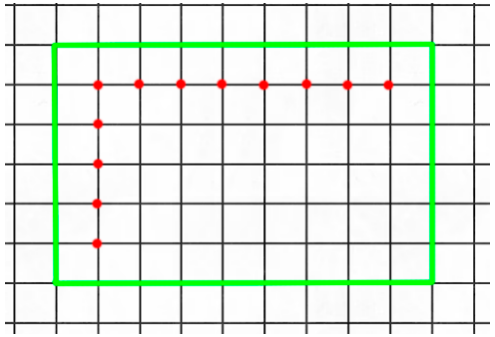
$x = 2, y = 2.4 \rightarrow$ 2 точки

$x = 3, y = 3.6 \rightarrow$ 3 точки

$x = 4, y = 4.8 \rightarrow$ 4 точки

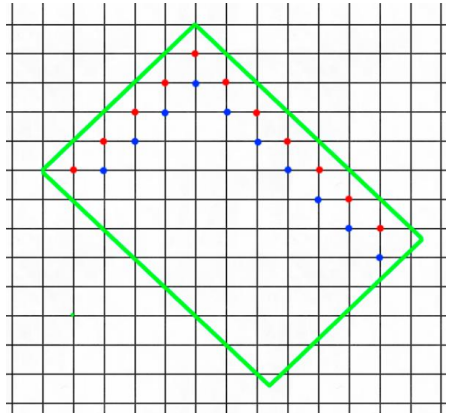
Прямоугольники

Для прямоугольников со сторонами, которые параллельны осям, подсчет самый простой - узнать количество точек по ширине, узнать количество точек по высоте и перемножить эти значения.

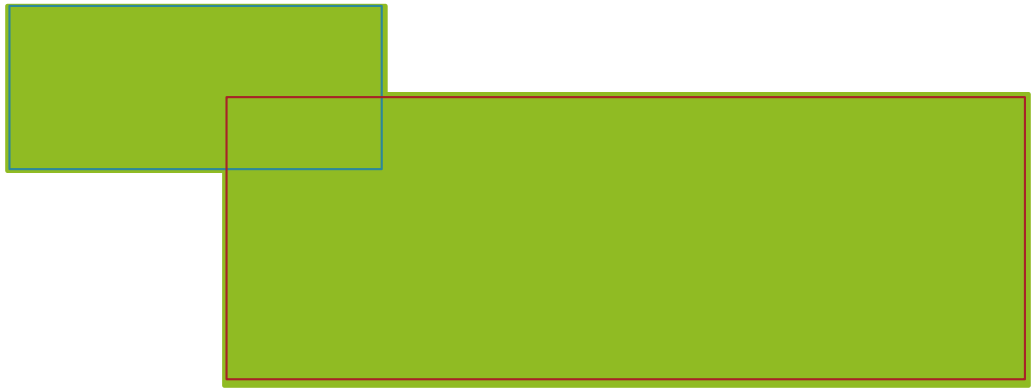


При наклоне прямоугольника на 45 градусов можно посчитать точки в двух соседних линиях, параллельных одной из сторон, и умножить их на общее количество.

Например, 5*7 + 4*7 = 63



Объединение фигур



Пересечение фигур

6

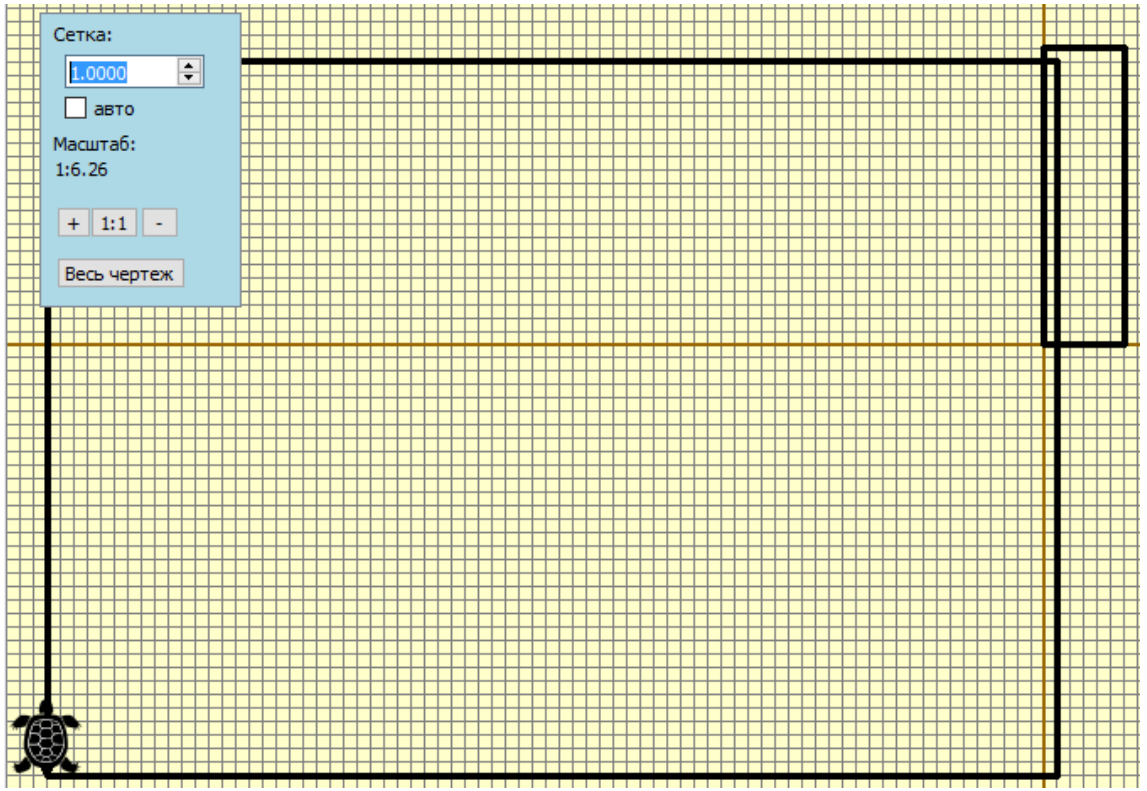
Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя существует 6 команд: **Поднять хвост**, означающая переход к перемещению без рисования; **Опустить хвост**, означающая переход в режим рисования; **Вперёд** *n* (где *n* – целое число), вызывающая передвижение Черепахи на *n* единиц в том направлении, куда указывает её голова; **Назад** *n* (где *n* – целое число), вызывающая передвижение в противоположном голове направлении; **Направо** *m* (где *m* – целое число), вызывающая изменение направления движения на *m* градусов по часовой стрелке, **Налево** *m* (где *m* – целое число), вызывающая изменение направления движения на *m* градусов против часовой стрелки.

Запись **Повтори** *k* **[Команда1 Команда2 … КомандаS]** означает, что последовательность из *S* команд повторится *k* раз.

Черепахе был дан для исполнения следующий алгоритм:

Повтори 9 [Вперёд 22 Направо 90 Вперёд 6 Направо 90]
Поднять хвост
Вперёд 1 Направо 90 Вперёд 5 Налево 90
Опустить хвост
Повтори 9 [Вперёд 53 Направо 90 Вперёд 75 Направо 90]

Определите периметр области пересечения фигур, ограниченных заданными алгоритмом линиями.



Пересечением фигур является прямоугольник со сторонами 1 и 21.
Периметр прямоугольника
1 + 1 + 21 + 21 = 44

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

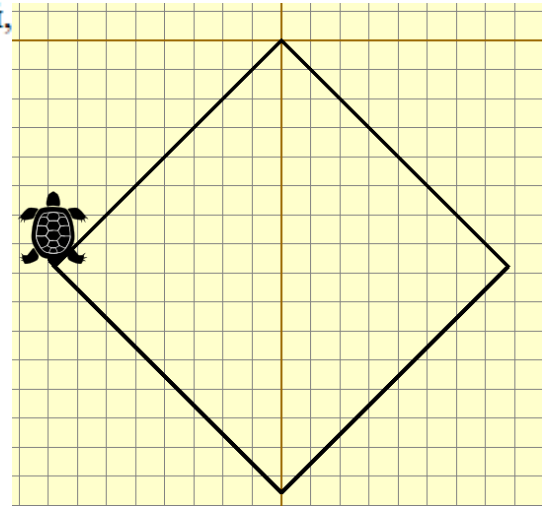
https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

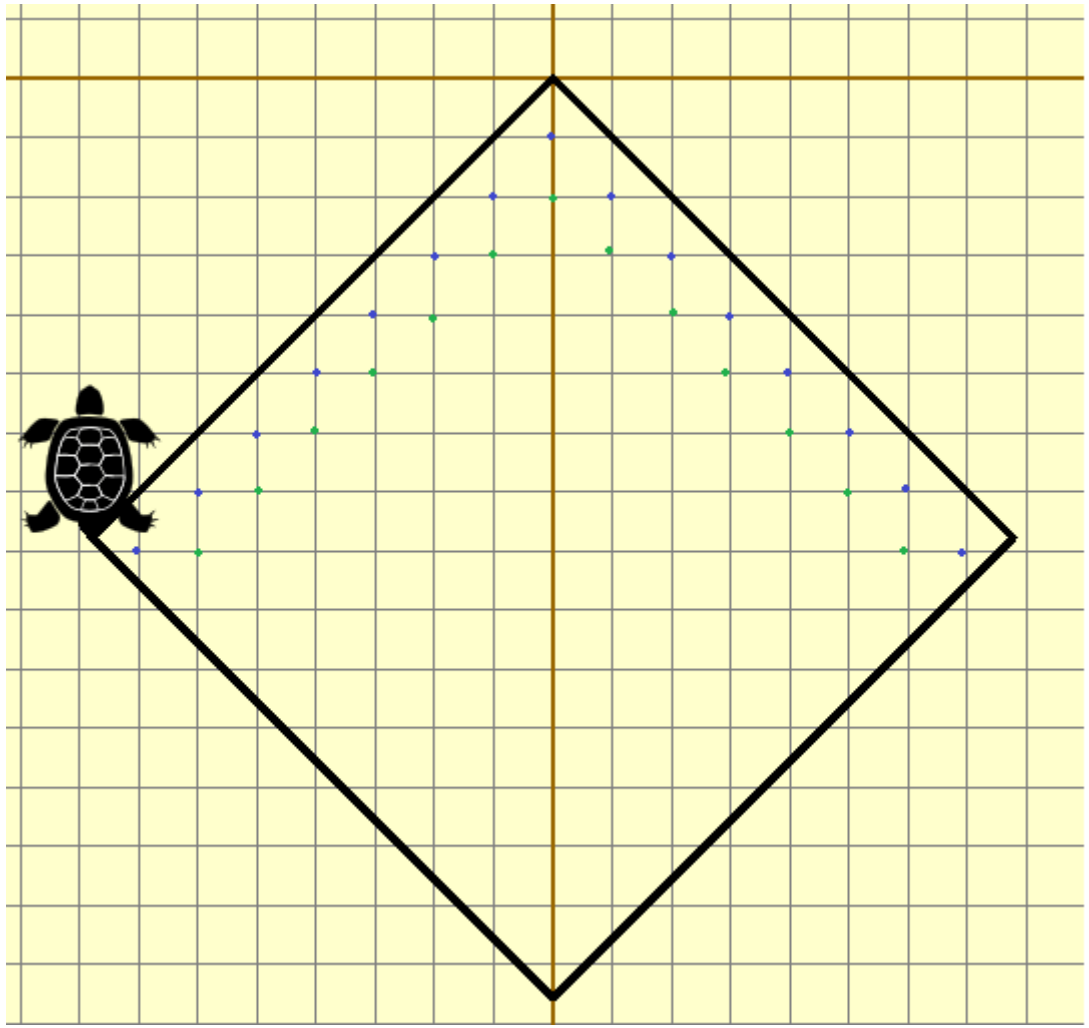
Задание 6

Исполнитель Черепаха действует на плоскости с декартовой системой координат. В начальный момент Черепаха находится в начале координат, её голова направлена вдоль положительного направления оси ординат, хвост опущен. При опущенном хвосте Черепаха оставляет на поле след в виде линии. В каждый конкретный момент известно положение исполнителя и направление его движения. У исполнителя есть две команды: **Вперёд** *n* (где *n* – целое число), вызывающая передвижение Черепахи на *n* единиц в том направлении, куда указывает её голова; **Направо** *m* (где *m* – целое число), вызывающая изменение направления движения на *m* градусов по часовой стрелке.
Запись **Повтори** *k* **[Команда1 Команда2 … КомандаS]** означает, что последовательность из *S* команд повторится *k* раз (где *k* – целое число).

Черепахе был дан для исполнения следующий алгоритм:
Направо 90 Повтори 7 [Направо 45 Вперёд 11 Направо 45].

Определите, сколько точек с целочисленными координатами находится внутри области, которая ограничена линией, заданной алгоритмом. Точки на линии учитывать не следует.

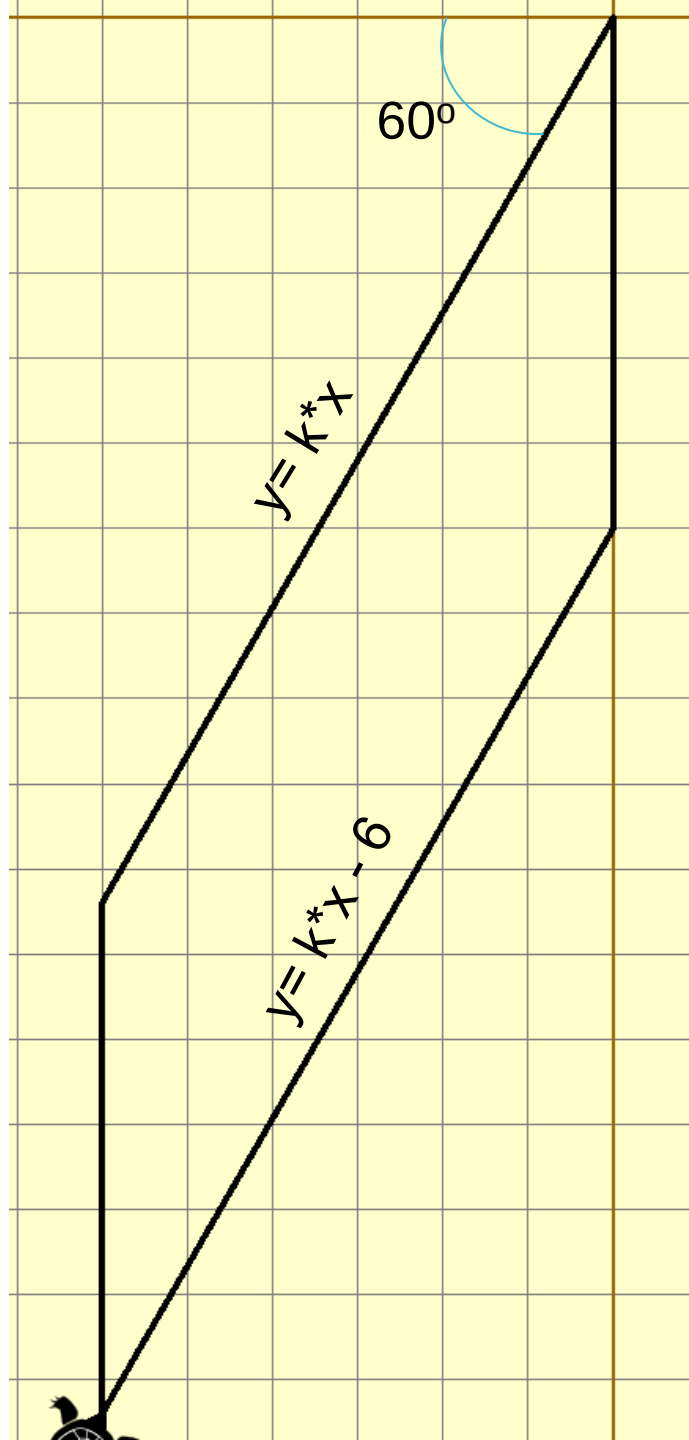




Синие: 8 x 8 = 64
Зеленые: 7 x 7 = 49
Всего: 64 + 49 = 113

192) (Досрочный ЕГЭ-2025)
Черепахе был дан для исполнения следующий алгоритм:
Направо 30
Повтори 3 [Направо 150 Вперёд 6 Направо 30 Вперёд 12]
Определите, сколько точек с целочисленными координатами будут находиться внутри области, которая ограничена линией, заданной алгоритмом. Точки на линии учитывать не следует.

Условие, по которому точка находится внутри фигуры:
$-6 < x < 0$, $k*x - 6 < y < k*x$

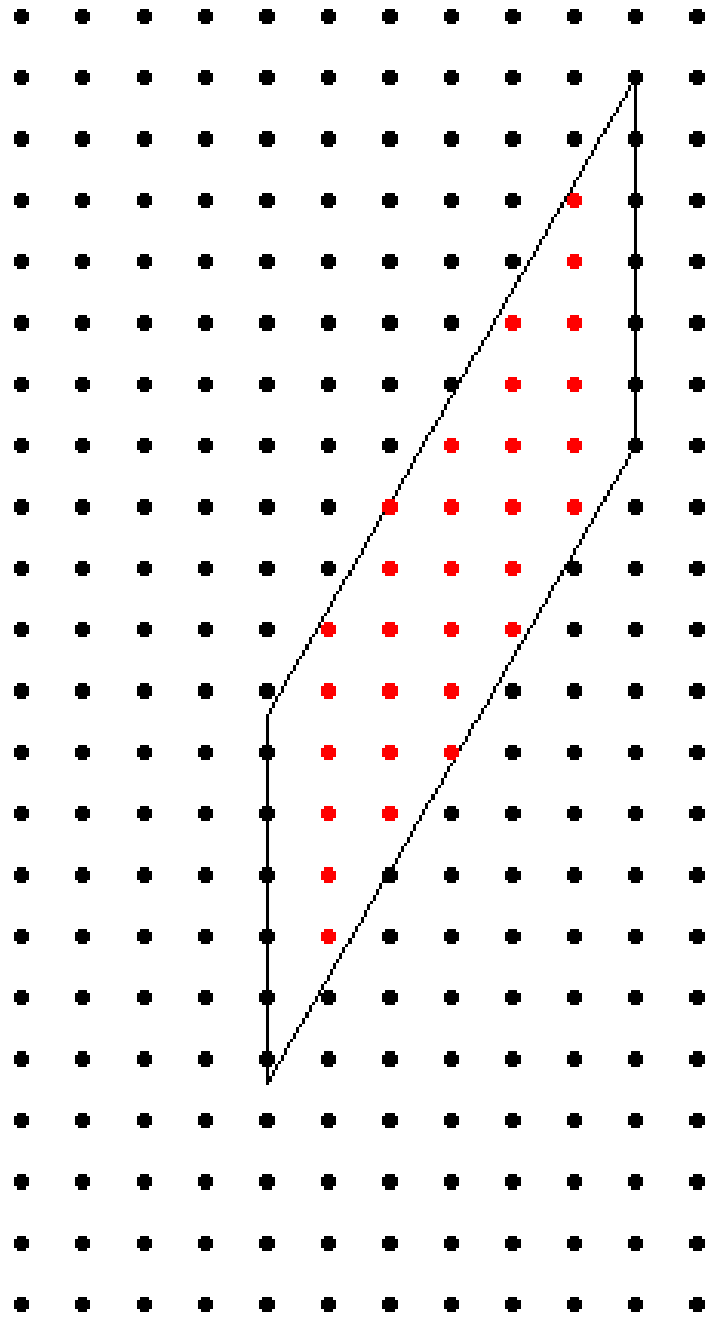


```
= RE
30  from turtle import *
    from math import *

    n = 20
    tracer(0)
    lt(90)
    pd()

    # фигура
    right(30)
    for _ in range(4):
        rt(150)
        fd(6*n)
        rt(30)
        fd(12*n)
    up()

    #подсчет точек внутри контура
    count = 0 # количество точек, попавших внутрь
    k = tan(pi / 3)        # тангенс угла пи/3 радиан (угла 60 градусов)
    for x in range(-10, 2):
      for y in range(-20, 2):
        goto(x * n, y * n)
        if (k * x - 6 < y < k * x) and (-6 < x < 0):
          dot(5, 'red')    # красная точка (точка попала внутрь)
          count += 1
        else:
          dot(5, 'black') # черная точка (точка не попала внутрь)
    print(count)
```



Задание 12 проверяет умение исполнить алгоритм для конкретного исполнителя с фиксированным набором команд

Пример задания

Исполнитель Редактор получает на вход строку цифр и преобразовывает её. Редактор может выполнять две команды, в обеих командах v и w обозначают цепочки символов.

1. заменить (v, w)
2. нашлось (v)

Первая команда заменяет в строке первое слева вхождение цепочки v на цепочку w. Если цепочки v в строке нет, эта команда не изменяет строку.

Вторая команда проверяет, встречается ли цепочка v в строке исполнителя Редактор.

Определите количество нулей в строке, получившейся в результате применения приведенной ниже программы к входной строке $1\underbrace{00..00}_{90}$, т.е. к строке, состоящей из 1, за которой следует 90 нулей подряд. В ответе запишите количество 0 в полученной строке.

```
НАЧАЛО
ПОКА нашлось (1)
    ЕСЛИ нашлось (10)
        ТО заменить (10, 0001)
        ИНАЧЕ заменить (1, 000)
    КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Алгоритм — это конечный набор точно определённых правил или инструкций, предназначенных для решения определённого класса задач. Эти инструкции описывают последовательность действий, которые должен выполнить исполнитель.

Существуют три основных **типа алгоритмов**:
•Линейные алгоритмы — команды выполняются последовательно, без каких-либо условий.
•Разветвляющиеся алгоритмы — последовательность действий зависит от выполнения определённых условий.
•Циклические алгоритмы — некоторые команды повторяются многократно, что называется циклом. При этом повторения выполняются ограниченное количество раз, чтобы алгоритм не зацикливался, нарушая требование результативности.

Типы задач для исполнителя «Редактор»:
1.Определение итоговой строки после выполнения команд исполнителя.
2.Определение начальной строки по известной итоговой строке.
3.Анализ итоговой строки после выполнения команд (например, подсчёт суммы цифр, проверка делимости, поиск простых чисел).

Способы решения подобных задач:
•Аналитический
•Алгебраический
•С использованием языка программирования
•С помощью электронных таблиц

Цикл while
позволяет выполнять блок кода многократно, пока выполняется определённое условие. Внутри цикла задаётся условие, которое проверяется перед каждым новым повторением. Как только условие перестаёт выполняться, цикл завершается.

Оператор if
проверяет, выполняется ли какое-то условие, и, если оно истинно (то есть True), выполняет блок кода. Это позволяет программе принимать решения на основе условий.

Метод replace
используется для замены одной подстроки на другую в строке.

Метод count
используется для подсчёта количества вхождений определённого элемента в строке или списке.

12 Исполнитель Редактор получает на вход строку цифр и преобразовывает её. Редактор может выполнять две команды, в обеих командах *v* и *w* обозначают цепочки цифр.

А) **заменить** (*v*, *w*).

Эта команда заменяет в строке первое слева вхождение цепочки *v* на цепочку *w*. Например, выполнение команды

заменить (111, 27)

преобразует строку 05111150 в строку 0527150.

Если в строке нет вхождений цепочки *v*, то выполнение команды **заменить** (*v*, *w*) не меняет эту строку.

Б) **нашлось** (*v*).

Эта команда проверяет, встречается ли цепочка *v* в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение «истина», в противном случае возвращает значение «ложь». Строка исполнителя при этом не изменяется.

Цикл

ПОКА *условие*
последовательность команд
КОНЕЦ ПОКА

выполняется, пока условие истинно.

В конструкции

ЕСЛИ *условие*
ТО *команда1*
КОНЕЦ ЕСЛИ

выполняется *команда1* (если условие истинно).

В конструкции

ЕСЛИ *условие*
ТО *команда1*
ИНАЧЕ *команда2*
КОНЕЦ ЕСЛИ

выполняется *команда1* (если условие истинно) или *команда2* (если условие ложно).

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 81 идущей подряд цифры 1? В ответе запишите полученную строку.

```
НАЧАЛО
ПОКА нашлось (11111) ИЛИ нашлось (888)
    ЕСЛИ нашлось (11111)
        ТО заменить (11111, 88)
        ИНАЧЕ заменить (888,8)
    КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 81 идущей подряд цифры 1? В ответе запишите полученную строку.

НАЧАЛО
ПОКА **нашлось** (11111) ИЛИ **нашлось** (888)
    ЕСЛИ **нашлось** (11111)
        ТО **заменить** (11111, 88)
        ИНАЧЕ **заменить** (888,8)
    КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ

```
====
881
>
```

```
s = 81*'1'

while '11111' in s or '888' in s:
    if '11111'in s:
        s = s.replace('11111','88',1)
    else:
        s = s.replace('888','8',1)
print(s)
```

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Дана программа для Редактора:

```
НАЧАЛО
ПОКА нашлось (19) ИЛИ нашлось (399) ИЛИ нашлось (999)
  ЕСЛИ нашлось (19)
    ТО заменить (19, 9)
  КОНЕЦ ЕСЛИ
  ЕСЛИ нашлось (399)
    ТО заменить (399, 91)
  КОНЕЦ ЕСЛИ
  ЕСЛИ нашлось (999)
    ТО заменить (999, 3)
  КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

```
====
46  for n in range(4,10000):
        s = '1' + n*'9'
        while '19' in s or '399' in s or '999' in s:
            s = s.replace('19','9',1)
            s = s.replace('399','91',1)
            s = s.replace('999','3',1)
##      s_d = 0
##      for x in s:
##          s_d += int(x)
        s_d = sum(int(x) for x in s)
        if s_d == 33:
            print(n)
            break
```

На вход приведённой выше программе поступает строка, начинающаяся с цифры «1», а затем содержащая n цифр «9» ($3 < n < 10\,000$).

Определите **наименьшее** значение n, при котором сумма цифр в строке, получившейся в результате выполнения программы, равна 33.

390) *(**О. Лысенков**)

Дана программа для Редактора:

```
ПОКА нашлось(4<) ИЛИ нашлось(11<) ИЛИ нашлось(00<)
  ЕСЛИ нашлось(11<)
    ТО заменить(11<, <9)
  КОНЕЦ ЕСЛИ
  ЕСЛИ нашлось(4<)
    ТО заменить(4<, <5)
  КОНЕЦ ЕСЛИ
  ЕСЛИ нашлось (00<)
    ТО заменить (00<,<92)
  КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
```

На вход приведённой ниже программе поступает строка, состоящая из 10 цифр из трехсимвольного набора 0, 4 или 1, расположенных в произвольном порядке, и идущего после них символа «<». Определите наибольшее возможное значение произведения числовых значений цифр в строке, которая может быть результатом выполнения программы.

На вход приведённой ниже программе поступает строка, состоящая из 10 цифр из трехсимвольного набора 0, 4 или 1, расположенных в произвольном порядке, и идущего после них символа «<». Определите наибольшее возможное значение произведения числовых значений цифр в строке, которая может быть результатом выполнения программы.

```
==== RE
9765625
```

```
from itertools import product
x = '041'
ans = []

for x in product(x, repeat = 10):
  s = ''.join(x) + '<'
  while '4<' in s or '11<' in s \
        or '00<' in s:
    s = s.replace('11<','<9',1)
    s = s.replace('4<','<5',1)
    s = s.replace('00<','<92',1)

  p = 1
  for a in s:
    if a != '<':
      p *= int(a)
  ans.append(p)
print(max(ans))
```

```
def alg( s ):
  while any( p in s for p in ['4<', '11<', '00<'] ):
    s = s.replace('11<', '<9', 1)
    s = s.replace('4<', '<5', 1)
    s = s.replace('00<', '<92', 1)
  return s

from math import prod
from itertools import product

pMax = 0
for w in product('014', repeat=10):
  s = alg( ''.join(w) + '<' )
  s = s.replace('<', '')
  p = prod( int(c) for c in s )
  pMax = max( p, pMax )

print( pMax )
```

https://kpolyakov.spb.ru/school/ege.htm

386) *Дана программа для Редактора:

ПОКА нашлось (111)
  заменить (111, 2)
  заменить (222, 11)
  заменить (1, 2)
КОНЕЦ ПОКА

Определите количество таких натуральных N из интервала [12345; 14567], для которых в результате применения данной программы к строке, состоящей из N единиц, получится строка, состоящая из трёх символов.

```
416 kol = 0
    for n in range(12345, 14568):
        s = n * '1'
        while '111' in s:
            s = s.replace('111','2',1).replace('222','11',1).replace('1','2',1)
        if len(s) == 3:
            kol += 1
    print(kol)
```

Задание 16 проверяет умение вычислять рекурсивные выражения

Пример задания
Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

F(n) = 1, при n = 1;
F(n) = n·F(n - 1) если n > 1.

Чему равно значение выражения (F(2024) / 4 + F(2023)) / F(2022)?

Типичные ошибки: решение задачи с помощью программирования рекурсии без математического упрощения, что приводит ошибке, связанной с ограничением на глубину рекурсии.

Для того, чтобы задать рекурсивную функцию, нужно определить

1. условие окончания рекурсии, то есть значения параметров функции, для которых значение функции известно или вычисляется без рекурсивных вызовов;
2. рекуррентную формулу (или формулы), с помощью которых значение функции для заданных значений параметров вычисляется через значение (или значения) функции для других значений параметров (то есть, с помощью рекурсивных вызовов)

Задачи, в которых требуется найти значение заданной рекурсивной функции при известных значениях параметров можно решать с помощью ручных вычислений, используя электронные таблицы или с помощью своей программы; последние два способа обычно более эффективны.

Чтобы избежать проблем с глубокой рекурсией, используется **мемоизация** — это метод оптимизации, используемый для ускорения программы, сохраняет результаты ранее вызванных функций и возвращает сохраненный результат при попытке вычислить ту же последовательность. Это также может называться кэшированием.

Суть метода — в создании структуры данных для запоминания уже рассчитанных значений функции.

16 Алгоритм вычисления значения функции $F(n)$, где n – *натуральное* число, задан следующими соотношениями:

$F(n) = 1$ при $n = 1$;

$F(n) = (n - 1) \times F(n - 1)$, если $n > 1$.

Чему равно значение выражения $(F(2024) + 2 \times F(2023)) / F(2022)$?

$$\frac{F(2024) + 2 \cdot F(2023)}{F(2022)} = \frac{2023 \cdot 2022 \cdot F(2022) + 2 \cdot 2022 \cdot F(2022)}{F(2022)} = \frac{2023 \cdot 2022 + 2 \cdot 2022}{1} = 4094550$$

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 16

Алгоритм вычисления значения функции $F(n)$, где n – *натуральное* число, задан следующими соотношениями:

$F(n) = n$ при $n \geq 2025$;

$F(n) = n \times 2 + F(n + 2)$, если $n < 2025$.

Чему равно значение выражения $F(82) - F(81)$?

```
1945
def f(n):
    if n >= 2025:
        return n
    else:
        return n*2+f(n+2)

print(f(82) - f(81))
```

```
1945
f = [0]*2028
for n in range(2027,0,-1):
    if n >= 2025:
        f[n] = n
    else:
        f[n] = n*2 + f[n+2]
print(f[82] - f[81])
```

236) (Досрочный ЕГЭ-2025) Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:
F(n) = 1 при n ≤ 5;
F(n) = n + F(n – 2), если n > 5.
Вычислите значение выражения F(2126) – F(2122).

$$F(2126) - F(2122) = 2126 + F(2124) - F(2122) = 2126 + 2124 + F(2122) - F(2122) = 4250$$

235) *(О. Лысенков) Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:
F(1) = 3;
F(n) = 5·F(n – 1), если n > 1.
Вычислите значение выражения F(10^{12}+10) / (25**(5·10^{11})), где ** означает возведение в степень.

$$\frac{F(10^{12}+10)}{25^{5\cdot 10^{11}}} = \frac{5^{10}\cdot F(10^{12})}{5^{10^{12}}} = \frac{5^{10}\cdot 5^{10^{12}-1}\cdot F(1)}{5^{10^{12}}} = \frac{5^{10}\cdot 5^{10^{12}-1}\cdot 3}{5^{10^{12}}} = 5^{10+10^{12}-1-10^{12}}\cdot 3 =$$

$$= 5^9 \cdot 3 = 5859375$$

Задание 17 проверяет умение составить алгоритм обработки числовой последовательности и записать его в виде простой программы

Пример задания
В файле содержится последовательность натуральных чисел. Элементы последовательности могут принимать целые значения от 1 до 100 000 включительно. Определите количество пар элементов последовательности, в которых остаток от деления хотя бы одного из элементов на 16 равен минимальному элементу последовательности. В ответе запишите количество найденных пар, затем максимальную из сумм элементов таких пар. В данной задаче под парой понимают два идущих подряд элемента последовательности.

Типичные ошибки: неверная формулировка условия в операторах ветвления и цикла, ошибки индексации, инициализации переменных, техническая ошибка – некорректный ввод из файла

Пути устранения ошибок в ходе обучения школьников: обращать внимание на результаты целочисленного деления отрицательных чисел, логические операции в языках программирования.

В задачах этого типа время выполнения несущественно для отрезков, заданных для перебора; поэтому можно использовать простой перебор без оптимизации.

Рекомендуется организовать ввод последовательности из файла и сохранить её в массиве; затем, двигаясь по массиву, определить условия отбора пар или троек элементов, и подсчитать количество пар (троек, отдельных элементов), удовлетворяющих этому условию.

ЕГЭ ПО ИНФОРМАТИКЕ
С ЕВГЕНИЕМ ДЖОБСОМ

Встроенные функции

Функция	Описание
max()/min()	Поиск максимального/ минимально числа
str()/int()	Перед в строку/число
abs()	Взятие модуля числа
sqrt() *(модуль math)* X**0.5	Взятие корня числа
gcd(a, b) *(модуль math)*	НОД чисел *a*, *b*
floor()/ceil() *(модуль math)*	Округление в меньшую/ большую сторону
int(a, b)	Перевод строки *a* из системы счисления *b* в десятичную

Полезности для 17 задания

Окончание числа (последняя цифра)

```
abs(x) % 10
```

Перебор **пар**/троек

```
for a, b in zip(a, a[1:]):
    ...
```

Количество совпадений (2)

```
def f(x): return условие
...
if f(a) + f(b) + f(c) == 2: ...
```

Длина числа (3)

```
len(str(abs(x))) == 3
```

Число содержит цифру (5)

```
'5' in str(x)
```

Сумма цифр

```
sum(map(int, str(abs(x))))
```

17 В файле содержится последовательность натуральных чисел. Её элементы могут принимать целые значения от 1 до 100 000 включительно. Определите количество пар последовательности, в которых остаток от деления хотя бы одного из элементов на 16 равен минимальному элементу последовательности. В ответе запишите количество найденных пар, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

```
=== RESTART:
1214 176024
```

```
f = open('demo_2025_17.txt')
p = [int(i) for i in f]
f.close()
M = 0
count = 0
M2 = 0

M = min(p)

for i in range(len(p) -1):
        s = p[i] + p[i+1]
        if abs(p[i])% 16 == M or abs(p[i + 1])% 16 == M:
            count += 1
            M2 = max(M2, s)

print(count, M2)
```

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 17

Задание выполняется с использованием прилагаемых файлов.

В файле содержится последовательность целых чисел. Её элементы могут принимать целые значения от −100 000 до 100 000 включительно. Определите количество троек последовательности, в которых все числа одного знака, при этом произведение минимального и максимального элементов тройки больше квадрата минимального элемента последовательности, который оканчивается на 15 и является трёхзначным числом. В ответе запишите количество найденных троек чисел, затем **минимальное** из произведений максимального и минимального элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

)7 863808

```
f = open('1_17.txt')
a = [int(x) for x in f]

m15 = min(x for x in a if abs(x) % 100 == 15 and \
          len(str(abs(x))) == 3)
ans = []
for x,y,z in zip(a, a[1:], a[2:]):
    m1 = max(x,y,z)
    m2 = min(x,y,z)
    if (x>0 and y>0 and z > 0 or x<0 and y<0 and z<0) \
       and m1*m2 > m15**2:
        ans.append(m1*m2)
print(len(ans), min(ans))
```

```
f = open('1_17.txt')
a = [int(x) for x in f]

m15 = min(x for x in a if abs(x) % 100 == 15 and \
          len(str(abs(x))) == 3)
ans = []
for i in range(len(a)-2):
    if (a[i]>0 and a[i+1]>0 and a[i+2]>0 or \
        a[i]<0 and a[i+1]<0 and a[i+2]<0) \
       and (max(a[i:i+3])*min(a[i:i+3])) > m15**2:
            ans.append(max(a[i:i+3])*min(a[i:i+3]))
print(len(ans), min(ans))
```

432) В файле **17-432.txt** содержится последовательность целых чисел, не превышающих по модулю 100 000. Определите количество троек элементов последовательности, в которых произведение максимального и минимального элементов тройки больше суммы всех отрицательных элементов последовательности. В ответе запишите количество найденных троек, затем абсолютное значение максимальной из сумм элементов таких троек. В данной задаче под тройкой подразумевается три идущих подряд элемента последовательности.

```
==========
10007 7953
```

```
f = open('17-432.txt')
a = [int(x) for x in f]

s_otr = sum(x for x in a if x < 0)
ans = []
for x,y,z in zip(a, a[1:],a[2:]):
    m1 = max(x,y,z)
    m2 = min(x,y,z)
    if m1*m2 > s_otr:
        ans.append(x+y+z)
print(len(ans), abs(max(ans)))
```

```
data = [ int(s) for s in open('17-432.txt') ]

s = sum( x for x in data if x < 0 )

N = len(data)
results = []
for i in range(N-2):
  triple = data[i:i+3]
  if min(triple)*max(triple) > s:
    results.append( sum(triple) )

print( len(results), abs(max(results)) )
```

https://kpolyakov.nichost.ru/school/ege.htm

Задание 23 проверяет умение анализировать ход исполнения алгоритма

Пример задания
Исполнитель преобразует число на экране.
У исполнителя есть две команды, которые обозначены латинскими буквами:
A. Вычти 1
B. Найди целую часть от деления на 2
Программа для исполнителя – это последовательность команд.
Сколько существует программ, для которых при исходном числе 30 результатом является число 1 и при этом траектория вычислений содержит число 12?

Типичные ошибки: в процессе рекуррентных вычислений выпускники забывают о том, что траектория обязана содержать или не содержать указанные в условии числа.

Варианты заданий:

- Подсчёт количества программ, преобразующих одно число в другое с использованием заданных команд
- С обязательным этапом – программа должна обязательно пройти через определённое число
- С избегаемым этапом – программа не должна проходить через указанное число
- С обязательным и запрещённым этапами одновременно – траектория должна содержать одно число и не содержать другое.

Динамическое программирование представляет собой способ оптимизации решения сложных задач, путём их последовательного разбиения на более мелкие и простые подзадачи (декомпозиция) и использования результатов решения этих подзадач.

Существует два подхода решения задач динамическим программированием:

- **Нисходящий подход** (решение задачи начинается с самой большой задачи, которая рекурсивно разбивается на подзадачи; для сохранения значения каждой подзадачи используется мемоизация).
- **Восходящий подход (**решение начинается с самых маленьких подзадач, и результаты используются для построения решения более крупных задач; такой подход реализуется с использованием итераций или табулирования).

23 Исполнитель преобразует число на экране.

У исполнителя есть две команды, которые обозначены латинскими буквами:

***A*. Вычти 2**

***B*. Найди целую часть от деления на 2**

Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 38 результатом является число 2 и при этом траектория вычислений содержит число 16?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы.

Например, для программы ***ABB*** при исходном числе 13 траектория состоит из чисел 11, 5, 2.

```
====
36
def f(x, y):
    if x == y:
        return 1
    if x < y:
        return 0
    else:
        return f(x - 2, y) + f (x // 2, y)

print (f(38, 16) * f(16, 2))
```

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 23

Исполнитель преобразует число на экране.

У исполнителя есть три команды, которые обозначены латинскими буквами:

***A*. Прибавить 1**

***B*. Прибавить 2**

***C*. Умножить на 2**

Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 3 результатом является число 18, при этом траектория вычислений содержит число 14 и не содержит 8?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. *Например*, для программы *CBA* при исходном числе 7 траектория будет состоять из чисел 14, 16, 17.

```
360
def f(x, y):
    if x == y:
        return 1
    if x > y or x == 8:
        return 0
    else:
        return f(x + 1, y) + f (x + 2, y)+ f (x * 2, y)

print (f(3, 14)*f(14, 18))
```

329) *(М. Шагитов, П. Хаматов) У исполнителя имеются две команды, которые обозначены латинскими буквами:

A. Прибавить 1

B. Прибавить сумму всех делителей

Первая команда увеличивает число на 1, вторая – увеличивает число на сумму всех его натуральных делителей (включая 1 и само число). Сколько существует программ, для которых при исходном числе 2 результатом является число 62?

```
=====
207
```

```
def s_div(n):
    s = 0
    for i in range(1, n+1):
        if n%i == 0:
            s = s+i
    return s

def f(x, y):
    if x == y:
        return 1
    if x > y :
        return 0
    else:
        return f(x + 1, y) + f (x + s_div(x), y)

print (f(2,62))
```

```
def sumdiv( n ):
  return sum( i for i in range(1,n+1)
                  if n % i == 0 )

def F( start, end ):
  return 1 if start == end else \
         0 if start > end else \
         F(start+1, end) + F(start+sumdiv(start), end)

print( F(2,62) )

#-----------------------------------
# Автор: М. Шагитов

lst = [0] * 300
lst[2] = 1
for i in range(2, 62):
    lst[i + 1] += lst[i]
    lst[i + sumdiv(i)] += lst[i]
print(lst[62])
```

Задание 24 проверяет умение создавать собственные программы (10–20 строк) для обработки символьной информации

Пример задания
Текстовый файл состоит из символов A, B, C, D, E и F. Определите в прилагаемом файле количество идущих подряд символов, среди которых пара AB (в указанном порядке) встречается ровно 100 раз. Для выполнения этого задания следует написать программу.

Типичные ошибки: неверно используют методы replace и split; не проверяют решение на небольших примерах; ошибки смыслового чтения в формулировке задания.

Разбиение строки

Разбиение строки происходит с учетом требований условия задачи, например, подстроки должны состоять из определенных символов и иметь максимально возможную длину.

```
with open("k7.txt", "r") as F:
  s = F.readline()
s = s.replace('A',' ').replace('C',' ')
s.split()
print(max(len(c) for c in s))
```

Если подстрока не должна содержать определенное буквосочетание, то строку разбиваем внутри этого буквосочетания.

```
with open("24.txt", "r") as F:
  s = F.readline()
s = s.replace('AB','A B')
s.split()
print(max(len(c) for c in s))
```

```
with open("24.txt", "r") as F:
  s = F.readline()
s = s.replace('ABCD','ABC BCD')
s.split()
print(max(len(c) for c in s))
```

Если в подстроке не должно содержаться больше одного определенного символа.

```
with open("24.txt", "r") as F:
  s = F.readline()
s = s.split('A')
mx = 0
for i in range(len(s)-1):
    c = s[i]+'A'+s[i+1]
    mx = max(mx, len(c))
print(mx)
```

Динамический подсчет

Строка (s)	A	A	A	*	B	A	B	B	A	A	*	B
Счетчик подходящих символов (m)	1	2	3	0	1	2	3	4	5	6 max	0	1

```
with open('24.txt','r') as F:
    s = F.readline()
m=[0]*len(s)
for i in range(len(s)):
    if s[i] in 'AB':
        m[i] = m[i-1] + 1
print(max(m))
```

Максимальное количество идущих подряд символов, среди которых каждые два соседних различны.

Строка (s)	A	B	C	C	B	A	B	B	A	A
Счетчик подходящих символов (m)	1	2	3	1	2	3	4 max	1	2	1

```
with open("24.txt", "r") as F:
  s = F.readline()
m = [1]*len(s)
for i in range(1,len(s)):
    if s[i] != s[i-1]:
        m[i] = m[i-1] + 1
print(max(m))
```

Метод двух указателей

Две позиции в строке отметим левым и правым указателем для выделения некоторой подстроки. Когда правый указатель увеличивается на 1, перемещаясь по строке вправо, левый указатель сдвигается в тот момент, когда происходит нарушение некоторых условий.

(№ 4524) Текстовый файл 24-181.txt содержит строку из заглавных латинских букв и точек, всего не более 10^6 символов. Определите максимальное количество идущих подряд символов, среди которых не более одной точки.

Строка (s)	A	B	C	.	D	A	B	E	G	H	.	B
Левый указатель (l)	↑											
Правый указатель (r)	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	
Левый указатель (l)					↑							
Правый указатель (r)											↑	↑

```
with open("24.txt", "r") as F:
    s = F.readline()
l = 0
mx = 0
k = 0
for r in range(len(s)):
    if s[r]=='.': k += 1
    while k > 1:
        if s[l] =='.': k -= 1
        l += 1
    mx = max(mx,r-l+1)
print(mx)
```

Регулярные выражения

Это строка, задающая шаблон поиска подстрок в тексте.

https://habr.com/ru/articles/349860/

Квантификаторы (указание количества повторений)

Шаблон	Описание	Пример	Применяем к тексту
`{n}`	Ровно n повторений	`\d{4}`	1, 12, 123, 1234, 12345
`{m, n}`	От m до n повторений включительно	`\d{2,4}`	1, 12, 123, 1234, 12345
`{m,}`	Не менее m повторений	`\d{3,}`	1, 12, 123, 1234, 12345
`{,n}`	Не более n повторений	`\d{,2}`	1, 12, 123
`?`	Ноль или одно вхождение, синоним `{0,1}`	`валы?`	вал, валы, валов
`*`	Ноль или более, синоним `{0,}`	`су\d*`	СУ, СУ1, СУ12, ...
`+`	Одно или более, синоним `{1,}`	`a\)+`	a), a)), a))), ba)])

Шаблоны, соответствующие одному символу

Во всех примерах соответствия регулярному выражению выделяются бирюзовым цветом с подчёркиванием.

Шаблон	Описание	Пример	Применяем к тексту
`.`	Один любой символ, кроме новой строки `\n`.	`м.л.ко`	молоко, малако, Им0л0коИхлеб
`[..]`	Один из символов в скобках, а также любой символ из диапазона `a-b`	`[0-9]` `[0-9A-Fa-f]`	12, 1F, 4B
`[^..]`	Любой символ, кроме перечисленных	`<[^>]>`	<1>, <a>, <>

Скобочные группы (?:...) и перечисления |

Некоторая строка подходит к регулярному выражению A|B тогда и только тогда, когда она подходит хотя бы к одному из регулярных выражений A или B.

Скобки (?:...) позволяют локализовать часть шаблона, внутри которого происходит перечисление.

Если в шаблоне регулярного выражения встречаются скобки (...) без ?:, то они становятся группирующими. Скобки (?=…) – поиск всех возможных совпадений.

Функции модуля re в Python

Функция	Её смысл
`re.search(pattern, string)`	Найти в строке `string` первую строчку, подходящую под шаблон `pattern`;
`re.fullmatch(pattern, string)`	Проверить, подходит ли строка `string` под шаблон `pattern`;
`re.split(pattern, string, maxsplit=0)`	Аналог `str.split()`, только разделение происходит по подстрокам, подходящим под шаблон `pattern`;
`re.findall(pattern, string)`	Найти в строке `string` все непересекающиеся шаблоны `pattern`;
`re.finditer(pattern, string)`	Итератор по всем непересекающимся шаблонам `pattern` в строке `string` (выдаются `match`-объекты);
`re.sub(pattern, repl, string, count=0)`	Заменить в строке `string` все непересекающиеся шаблоны `pattern` на `repl`;

Если функции **re.search, re.fullmatch** не находят соответствие шаблону в строке, то они возвращают None, функция **re.finditer** не выдаёт ничего. Однако если соответствие найдено, то возвращается match-объект.

Метод	Описание	Пример
`match[0]`, `match.group()`	Подстрока, соответствующая шаблону	`match = re.search(r'\w+', r'$$ What??') match[0] # -> 'What'`
`match.start()`	Индекс в исходной строке, начиная с которого идёт найденная подстрока	`match = re.search(r'\w+', r'$$ What??') match.start() # -> 3`
`match.end()`	Индекс в исходной строке, который следует сразу за найденной подстрока	`match = re.search(r'\w+', r'$$ What??') match.end() # -> 7`

24

Текстовый файл состоит из цифр 0, 6, 7, 8, 9 и знаков арифметических операций «–» и «*» (вычитание и умножение). Определите максимальное количество символов в непрерывной последовательности, которая является корректным арифметическим выражением с целыми неотрицательными числами. В этом выражении никакие два знака арифметических операций не стоят рядом, в записи чисел отсутствуют незначащие (ведущие) нули и число 0 не имеет знака.

В ответе укажите количество символов.

```
154
from re import *
s = open('demo_2025_24.txt').readline()
n = r'(?:0|[6789][06789]*)'
reg = rf'{n}([*-]{n})+'
m = max([x.group() for x in finditer(reg,s)],key = len)
print(len(m))
```

```
154
from re import *
s = open('demo_2025_24.txt').readline()
a = findall(r'(?:0|[6789][06789]*)(?:[*-](?:0|[6789][06789]*))*',s)
m = max(a, key = len)
print(len(m))
```

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 24

Задание выполняется с использованием прилагаемых файлов.

Текстовый файл состоит из *десятичных цифр* и *заглавных букв латинского алфавита*. Определите в этом файле последовательность идущих подряд символов, представляющих собой запись максимального чётного 14-ричного числа. В ответе запишите количество символов (значащих цифр в записи числа) в этой последовательности.
Примечание. Латинские буквы *A*, *B*, *C* и *D* означают цифры из алфавита 14-ричной системы счисления.

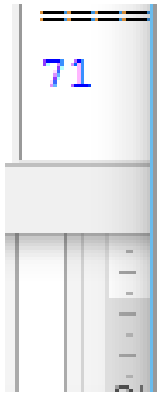
2598

```
from re import *
s = open('1_24.txt').readline()
reg = r'([1-9A-D][0-9A-D]*[02468AC])'

m = max([x.group() for x in finditer(reg,s)],key = len)
print(len(m))
```

344) (К. Багдасарян) Текстовый файл 24-337.txt состоит не более чем из 106 символов и содержит десятичные цифры и заглавные буквы латинского алфавита. Определите максимальное количество символов в непрерывной последовательности, которые могут представлять запись натурального числа в четырнадцатеричной системе счисления, которое начинается с 1 и кратно 7. Цифры, числовое значение которых превышает 9, обозначены латинскими буквами, начиная с буквы A. Гарантируется наличие такой последовательности.

Числа в 10сс, кратные 7	Последняя цифра в 14сс
7	7
14	0
21	7
28	0
35	7
42	0
…	…



```
71
```

```
from re import *
s = open('24-337.txt').readline()
reg = r'1[0-9A-D]*(?:0|7)'

m = max([x.group() for x in finditer(reg,s)],key = len)
print(len(m))
```

344) (К. Багдасарян) Текстовый файл 24-337.txt состоит не более чем из 106 символов и содержит десятичные цифры и заглавные буквы латинского алфавита. Определите максимальное количество символов в непрерывной последовательности, которые могут представлять запись натурального числа в четырнадцатеричной системе счисления, которое начинается с 1 и кратно 7. Цифры, числовое значение которых превышает 9, обозначены латинскими буквами, начиная с буквы A. Гарантируется наличие такой последовательности.

```
from re import findall

s = open('24-337.txt').readline()

num7 = r'1[0-9A-D]*(?:0|7)'
pattern = fr'(?=({num7}))'

parts = findall( pattern, s )
sMax = max( parts, key = len )
print( len(sMax), sMax)

# Автор: К. Багдасарян

s = open('24-337.txt').readline()

sub = sMax = ''
for c in s:
    if (c in '0123456789ABCD' and sub) or c == '1':
        sub += c
        if c in '07':
            if len(sub) > len(sMax):
                sMax = sub
    else:
        sub = ''
print( len(sMax), sMax )
```

https://kpolyakov.nichost.ru/school/ege.htm

Задание 25 проверяет умение создавать собственные программы (10–20 строк) для обработки целочисленной информации

Пример задания
Пусть M – сумма минимального и максимального натуральных делителей целого числа, не считая единицы и самого числа. Если таких делителей у числа нет, то значение M считается равным нулю. Напишите программу, которая перебирает целые числа, большие 700 000, в порядке возрастания и ищет среди них такие, для которых значение M оканчивается на 4. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения M.
Типичные ошибки: неправильное понимание того, что именно нужно найти (делитель, число, сумму и т. д.) и каким критериям должен соответствовать результат; выбор неэффективного или некорректного алгоритма поиска делителей или выполнения других необходимых операций. Например, перебор делителей до N вместо sqrt(N) приводит к существенному замедлению.

Перебрать все целые числа на отрезке [a; b] и подсчитать, для скольких из них выполняется некоторое условие (общая структура)

```
count = 0
for n in range(a, b+1):
  if условие выполнено:
    count += 1
print( count )
```

Проверить делимость числа *n* на число *d* можно с помощью операции взятия остатка от деления *n* на *x*: если остаток равен 0, число *n* делится на *x* нацело

```
if n % d == 0:
      print("Делится")
else: print("Не делится")
```

Определить число делителей натурального числа *n* можно в цикле, в котором перебираются все возможные делители *d* от 1 до *n*, при обнаружении делителя увеличивается счётчик делителей:

```
count = 0
for d in range(1, n+1):
  if n % d == 0:
    count += 1
print( count )
```

Если требуется определить не только количество делителей, но и сами делители, нужно сохранять их в массиве.

```
divs = []
for d in range(1,n+1):
  if n % d == 0:
    divs.append(d)
```

Определить простое число можно, считая общее количество его делителей; если их ровно два, то число простое, если не два – не простое:

```
nDel = 0
for d in range(1, n+1):
    if n % d == 0:
        nDel += 1
if nDel == 2:
    print( "Число простое" )
else:
    print( "Число составное" )
```

Если уже найдено больше двух делителей, то число не простое и можно досрочно закончит работу цикла с помощью оператора break.

```
for d in range(1, n+1):
  if n % d == 0:
    nDel += 1
    if nDel > 2:
      break
```

Другой вариант – считать количество делителей числа на отрезке [2; *n*–1]; как только хотя бы один такой делитель будет найден, можно завершить цикл, потому что число явно не простое.

Можно сделать то же самое с помощью логической переменной:

```
prime = True
for d in range(2, n):
    if n % d == 0:
        prime = False       # уже не простое
        break       # досрочный выход из цикла
if prime:
    print( "Число простое" )
else:
    print( "Число составное" )
```

25

Пусть M – сумма минимального и максимального натуральных делителей целого числа, не считая единицы и самого числа. Если таких делителей у числа нет, то считаем значение M равным нулю.
Напишите программу, которая перебирает целые числа, бóльшие 800 000, в порядке возрастания и ищет среди них такие, для которых M оканчивается на 4. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – соответствующие им значения M.

Например, для числа 20 $M = 2 + 10 = 12$.

Количество строк в таблице для ответа избыточно.

```
==============
800004 400004
800009 114294
800013 266674
800024 400014
800033 61554
```

File Edit Format Run Options Window Help

```
x = 800001
count = 0

while count <5:
    for d in range (2, int(x**0.5)+1):
        if x % d == 0:
            M = d + x //d
            break

    if M % 10 ==4:
            print (x, M)
            count +=1
    x += 1
```

https://docs-python.ru/standart-library/modul-fnmatch-python/

Модуль **fnmatch** обеспечивает поддержку подстановочных знаков в стиле оболочки Unix, которые не совпадают с регулярными выражениями.

Специальные символы, используемые в подстановочных знаках в стиле оболочки:

Pattern	Meaning
*	Соответствует всему
?	Соответствует любому отдельному символу
[seq]	Соответствует любому символу в скобках
[!seq]	Соответствует любому символу не в скобках

Для буквального совпадения заключите метасимволы в скобки. Например, [?] Соответствует символу ?.

Разделитель имени файла '/' в Unix не является специальным для этого модуля.

Функция fnmatch() модуля fnmatch проверяет, соответствует ли строка имени файла шаблонной строке, возвращая `True` или `False`.

Функция fnmatchcase() модуля fnmatch проверяет, соответствует ли имя файла `filename` строке шаблона `pattern`, возвращая `True` или `False`. Сравнение чувствительно к регистру и к строке `filename`.

Функция filter() модуля fnmatch вернет подмножество списка имен `filename`, которые соответствуют шаблону `pattern`.

Функция translate() модуля fnmatch возвращает шаблон имени файла в стиле Unix, преобразованный в регулярное выражение для использования с функцией `re.match()`.

Среди натуральных чисел, не превышающих 10^9, найдите все числа, соответствующие маске 23?456?89 и делящиеся на 17 без остатка. В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце – соответствующие им частные от деления на 17.

```
from fnmatch import *
for x in range(0,10**9,17):
    if fnmatch(str(x),'23?456?89'):
        print(x, x//17)
```

```
231456989 13615117
232456589 13673917
233456189 13732717
236456689 13909217
237456289 13968017
```

Например, маске 123*4?5 соответствуют числа 123405 и 12300425. Среди натуральных чисел, не превышающих 10^9, найдите все числа, соответствующие маске 123*567? и делящиеся на 169 без остатка. В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце — соответствующие им частные от деления на 169.

```
from fnmatch import *
for x in range(0,10**9,169):
    if fnmatch(str(x),'123*567?'):
        print(x,x//169)
```

```
==== RESTART: C:/
12325677 72933
12385672 73288
```

```
# * - нет цифр
for a in '0123456789':
    x = int(f'123567{a}')
    if x%169==0:
        print(x,x//169)

# * - 1 цифра
for a in '0123456789':
    for b in '0123456789':
        x = int(f'123{a}567{b}')
        if x%169==0:
            print(x,x//169)

# * - 2 цифры
for a in '0123456789':
    for b in '0123456789':
        for c in '0123456789':
            x = int(f'123{a}{b}567{c}')
            if x%169==0:
                print(x,x//169)
```

```
File  Edit  Shell  Debug
Python 3.8.3 (tags/
tel)] on win32
Type "help", "copyr
>>>
==== RESTART: C:/Us
12325677 72933
12385672 73288
123165679 728791
123225674 729146
123515678 730862
123575673 731217
123865677 732933
123925672 733288
>>>
```

(№ 5281) (А. Агафонцев) Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:
— символ «?» означает ровно одну произвольную цифру;
— символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.
Найдите наименьшие 7 чисел, удовлетворяющих маске ?6*6*?6 и при этом кратных 6, 7 и 8. Выведите эти числа в порядке возрастания, справа от каждого числа выведите сумму его делителей.

```
from fnmatch import *

def div(x):
    d=set()
    for i in range(1,int(x**0.5)+1):
        if x%i==0:
            d.add(i)
            d.add(x//i)
    return sorted(d)


for x in range(int(10**6)):
    if fnmatch(str(x),'?6*6*?6') and x%6==0 and x%7==0 and x%8==0:
        print(x,sum(div(x)))
```

```
==== RESTART: C:
56616 162240
66696 191040
161616 527744
166656 523264
266616 862680
360696 1094400
366576 1083264
460656 1476592
465696 1723680
466536 1333440
560616 1658880
565656 1819440
566496 1847664
665616 2007312
```

(№ 5032) (Б. Михлин) Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:
- символ «?» означает ровно одну произвольную цифру;
- символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Например, маске 123*4?5 соответствуют числа 123405 и 12300425. Найдите все натуральные числа, делящиеся нацело на BA_{16}, шестнадцатеричный код которых соответствует маске 1?DED?BABA. В ответе запишите найденные числа в десятичной системе счисления в порядке убывания, а справа от каждого числа – соответствующее частное от деления на BA_{16}.

```
for a in 'fedcba9876543210':
    for b in 'fedcba9876543210':
        x = int(f'1{a}ded{b}baba',16)
        ba16=11*16+10
        if x%ba16==0:
            print(x,x//ba16)
```

```
==== RESTART: C:/Users/s
132587371194 712835329
119702862522 643563777
106818353850 574292225
>>>
```

(№ 5620) (К. Багдасарян) Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:
- символ «?» означает ровно одну произвольную цифру;
- символ «Ч» означает ровно одну произвольную четную цифру;
- символ «Н» означает ровно одну произвольную нечетную цифру;

Например, маске Ч?Н2 соответствуют числа 2912, 6012, 6772 и т.д. Среди натуральных чисел, не превышающих 10^8 найдите все числа, соответствующие маске 11Ч??Н11, делящиеся на 2023 без остатка. В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце – соответствующие им результаты деления этих чисел на 2023.

```
from fnmatch import *
for x in range(0,10**8,2023):
    if fnmatch(str(x),'11[02468]??[13579]11'):
        print(x, x//2023)
```

```
==== RESTART: C
11039511 5457
11444111 5657
11848711 5857
>>>
```

ИЛИ

Назовём маской числа последовательность цифр, в которой также могут встречаться следующие символы:

- символ «?» означает ровно одну произвольную цифру;
- символ «*» означает любую последовательность цифр произвольной длины; в том числе «*» может задавать и пустую последовательность.

Например, маске 123*4?5 соответствуют числа 123405 и 12300405.

Среди натуральных чисел, не превышающих 10^{10}, найдите все числа, соответствующие маске 3?12?14*5, делящиеся на 1917 без остатка.

В ответе запишите в первом столбце таблицы все найденные числа в порядке возрастания, а во втором столбце – соответствующие им результаты деления этих чисел на 1917.

Количество строк в таблице для ответа избыточно.

```
================ REST
351261495 183235
3212614035 1675855
3412614645 1780185
3712414275 1936575
3912414885 2040905
```

```
from fnmatch import *
for x in range(0, 10**10 +1, 1917):
  if fnmatch(str(x),'3?12?14*5'):
    print(x, x // 1917)
```

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 25

Пусть R – сумма всех различных натуральных делителей целого числа.

Напишите программу, которая перебирает целые числа, бóльшие 500 000, в порядке возрастания и ищет среди них такие, для которых значение R оканчивается на цифру 6. В ответе запишите в первом столбце таблицы первые пять найденных чисел в порядке возрастания, а во втором столбце – пять соответствующих этим числам значений R.

Например, для числа 20 $R = 1 + 2 + 4 + 5 + 10 + 20 = 42$.

Количество строк в таблице для ответа избыточно.

```
====
36
6
6
6
96
```

```
def div(x):
    d = set()
    for i in range(1, int(x**0.5)+1):
        if x%i == 0:
            d |= {i, x//i}
    return sorted(d)

k = 0
for x in range(500001, 500100):
    d = div(x)
    if sum(d)%10 == 6:
        print(x, sum(d))
        k += 1
        if k == 5:
            break
```

Задание 26 проверяет умение обрабатывать целочисленную информацию с использованием сортировки

Пример задания

При онлайн-покупке билета на концерт известно, какие места в зале уже заняты. Необходимо купить два билета на такие соседние места в одном ряду, чтобы перед ними все кресла с такими же номерами во всех рядах были свободны, а ряд находился как можно дальше от сцены. Если в этом ряду таких пар мест несколько, найдите пару с наименьшими номерами. В ответе запишите два целых числа: искомый номер ряда и наименьший номер места в найденной паре.

Нумерация рядов и мест ведётся с 1. Гарантируется, что хотя бы одна такая пара в зале есть.

Входные данные. В первой строке входного файла находятся три числа: N – количество занятых мест в зале (целое положительное число, не превышающее 10000), M – количество рядов (целое положительное число, не превышающее 100 000) и K – количество, мест в каждом ряду (целое положительное число, не превышающее 100 000). В следующих N строках находятся пары натуральных чисел: номер ряда и номер места занятого кресла соответственно (первое число не превышает значения M, а второе - K).

Выходные данные. Два целых числа: наибольший номер ряда и наименьший номер места в найденной паре.

Чтение данных из файла

```
with open("26.txt") as F:
```

если в текущей строке файла находится целое число, то прочитать его в переменную x можно так:

```
x = int( F.readline() )
```

если в строке записаны два числа, после чтения (F.readline()) строку нужно разбить на отдельные части по пробелам между числами (каждая часть – символьная запись числа) и затем каждую часть преобразовать в целое число; например, чтение двух чисел:

```
s = F.readline()
st = s.split()
x, y = map( int, st )
или в компактной форме
x, y = map( int, F.readline().split() )
```

Хранение массива данных

В языке Python для хранения массива данных используется список:

```
data = [0]*N
with open("26.txt") as F:
  for i in range(N):
    data[i] = int( F.readline() )
```

или с помощью генератора списка

```
with open("26.txt") as F:
  data = [ int( F.readline() ) for i in range(N) ]
```

Сортировка массива

В языке Python для сортировки по возрастанию используется метод sort:

```
data.sort()
```

Для сортировки по убыванию в вызов метода добавляем именованный аргумент reverse со значением True:

```
data.sort( reverse = True )
```

Для сортировки по другому критерию (например, по последней цифре числа) добавляют именованный аргумент key, который указывает на функцию, вычисляющую нужно значение, например:

```
def lastDigit( n ):
  return n % 10
... # заполнение массива data
data.sort( key = lastDigit )
```

Простую функцию можно не оформлять как отдельную подпрограмму, а записать как неименованную функцию (лямбда-функцию) :

```
data.sort( key = lambda x: x % 10 )
```

Сортировка массива

Иногда данные в массиве представляют собой пары или тройки чисел, объединённые в кортежи. В этом случае при стандартной сортировке сначала сравниваются первые элементы кортежей, если они равны – вторые и т.д. Чтобы задать свой порядок сортировки, нужно использовать аргумент key с обычной функцией или лямбда-функцией. Например,

```
data.sort( key = lambda x: (-x[1], x[0]%10) )
```

В этом примере происходит сортировка по убыванию (знак «минус») второго числа в кортеже, x[1], а если вторые элементы равны - по возрастанию последней цифры первого элемента кортежа, x[0].

Если нужно создать новый массив, не изменяя исходные данные, используется функция sorted().
Её первый аргумент – массив, а остальные совпадают с аргументами метода sort. Например,

```
data1 = sorted( data, key = lambda x: (-x[1], x[0]%10) )
```

Ключ сортировки

`key` – ключ, по которому будет совершена сортировка

`lambda x` - лямбда-функция, которая будет ключом для сортировки

Сортировка по модулю

```
x = [1, -4, 7, 9, -23, -6]
x.sort(key=abs)
#[1, -4, -6, 7, 9, -23]
```

Сортировка по сумме цифр

```
x = [118, 46, 8, 94, 273, 76]
x.sort(key=lambda x: sum(map(int, str(x))))
#[8, 118, 46, 273, 94, 76]
```

Сортировка по убывания первого элемента и возрастанию второго

```
x = [[4, 9], [2, 6],[2, 3],[4, 7],[4, 8]]
x.sort(key=lambda x: (-x[0], x[1]))
#[[4, 7], [4, 8], [4, 9], [2, 3], [2, 6]]
```

Сортировка по возрастанию первого и второго элемента

```
x = [[4, 9], [2, 6],[2, 3],[4, 7],[4, 8]]
x.sort(key=lambda x: (x[1], x[0]))
#[[2, 3], [2, 6], [4, 7], [4, 8], [4, 9]]
```

26

Задание выполняется с использованием прилагаемых файлов.

Во время сессии студенты сдают 4 экзамена, за каждый из которых можно получить от 2 до 5 баллов. Студенты, получившие хотя бы одну «двойку», считаются не сдавшими сессию. Результаты сессии публикуются в виде рейтингового списка, в котором сначала указаны идентификационные номера студентов (ID), сдавших сессию, в порядке убывания среднего балла за сессию, а в случае равенства средних баллов – в порядке возрастания ID. Затем располагаются ID студентов, не сдавших сессию: сначала – получивших одну «двойку», затем – две «двойки», потом ID студентов с тремя «двойками» и, наконец, ID студентов, получивших по 2 балла за каждый из экзаменов. Если студенты имеют одинаковое количество «двоек», то их ID в рейтинге располагаются в порядке возрастания.

Повышенную стипендию получают студенты, занявшие в рейтинговом списке первые 25 % мест, при условии отсутствия у них «двоек». Гарантируется, что без «двоек» сессию сдали не менее 25 % студентов. Найдите ID студента, который занимает последнее место среди студентов с повышенной стипендией, а также ID первого в рейтинговом списке студента, который имеет более двух «двоек».

В ответе запишите два целых положительных числа: сначала ID студента, который занимает последнее место среди студентов с повышенной стипендией, затем ID первого в рейтинговом списке студента, который имеет более двух «двоек».

Входные данные

В первой строке входного файла находится число N, обозначающее количество студентов (целое положительное число, не превышающее 10 000). Каждая из следующих N строк содержит 5 чисел через пробел: ID студента (целое положительное число, не превышающее 100 000) и четыре оценки, полученные им за сессию. Гарантируется, что общее число студентов N кратно 4 и хотя бы один студент имеет более двух «двоек». Во входном файле все ID различны.

Задача решается с помощью сортировки в электронной таблице.

Для определения ID студента, который занимает последнее место среди студентов с повышенной стипендией:

1. Сортировка по возрастанию количеству двоек
2. Сортировка по убыванию среднего балла студентов
3. Сортировка по возрастанию ID

Для определения ID первого в рейтинговом списке студента, который имеет более двух «двоек»:

1. Сортировка по возрастанию количеству двоек
2. Сортировка по возрастанию ID
3. Фильтр по количеству «2»: >2

	A	B	C	D	E	F	G	H	I	J	K
1	1944	4	3	4	4	0	3,75			9964	
2	52099	4	5	3	5	0	4,25				
3	13216	5	3	4	5	0	4,25				
4	48820	3	4	2	4	1					
5	8155	3	2	5	3	1					
6	23829	5	3	4	3	0					
7	52527	3	3	4	5	0					
8	13906	4	3	5	5	0					
9	53780	3	3	5	5	0					
10	38366	4	3	4	4	0					
11	54144	3	3	4	5	0					
12	13447	5	3	3	5	0					
13	44508	3	4	3	4	0					
14	30811	3	3	4	2	1					
15	21252	3	5	3	3	0					
16	17933	2	4	2	3	2					
17	10343	4	4	3	4	0					
18	27547	5	4	4	3	0					
19	34930	5	4	4	5	0					
20	4707	5	4	3	3	0					
21	16157	3	3	5	5	0					
22	48802	4	5	4	5	0					
23	33952	2	3	3	5	1					
24	17418	3	3	5	4	0					
25	49200	3	2	2	3	2					
26	16629	3	3	5	5	0					
27	26640	3	5	4	5	0					
28	30527	3	5	4	2	1					
29	22193	5	5	4	5	0					
30	52325	5	5	3	5	0					
31	23335	4	5	4	5	0	4,5				

Сортировка

Условия сортировки | Параметры

Столбец: F — По возрастанию / По убыванию

Ключ сортировки 2

Столбец: G — По возрастанию / По убыванию

Ключ сортировки 3

Столбец: A — По возрастанию / По убыванию

Ключ сортировки 4

Столбец: - не определён - — По возрастанию / По убыванию

Параметры сортировки

Заголовки: Диапазон содержит метки столбцов

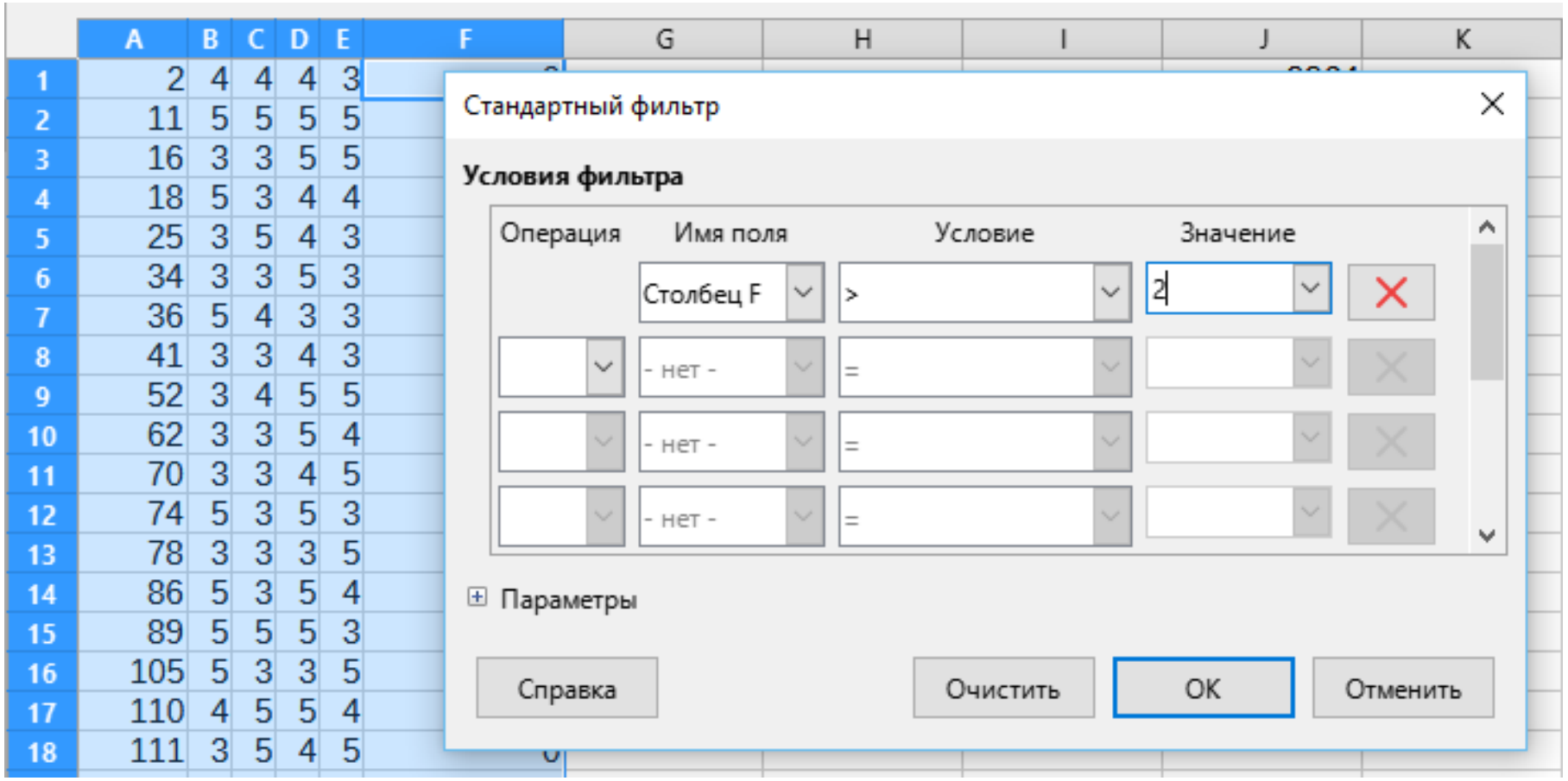
Направление: Сверху вниз (сортировка строк) / Слева направо (сортировка столбцов)

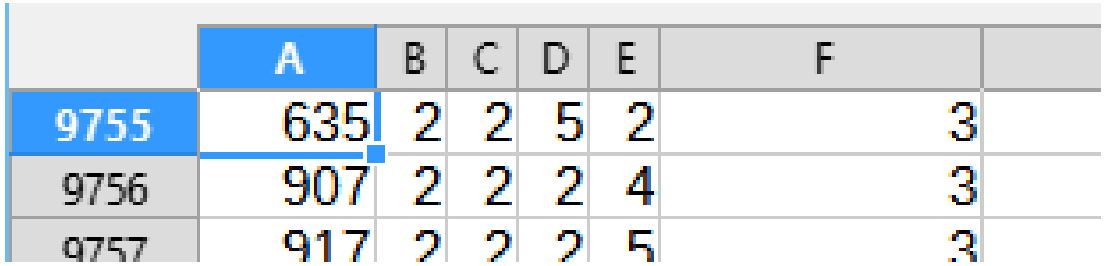
Справка | Восстановить | OK | Отменить

Номер последней строки среди 25% лучших студентов 0,25*9964 = 2491

J
9964
.=J1/4
.=A2491

J
9964
2491
52326


A B C D E F G H I J K
1 2 4 4 4 3
2 11 5 5 5 5
3 16 3 3 5 5
4 18 5 3 4 4
5 25 3 5 4 3
6 34 3 3 5 3
7 36 5 4 3 3
8 41 3 3 4 3
9 52 3 4 5 5
10 62 3 3 5 4
11 70 3 3 4 5
12 74 5 3 5 3
13 78 3 3 3 5
14 86 5 3 5 4
15 89 5 5 5 3
16 105 5 3 3 5
17 110 4 5 5 4
18 111 3 5 4 5
Стандартный фильтр
Условия фильтра
Операция
Имя поля
Условие
Значение
Столбец F
>
2
- нет -
=
Параметры
Справка
Очистить
OK
Отменить


A B C D E F
9755 635 2 2 5 2 3
9756 907 2 2 2 4 3
9757 917 2 2 2 5 3

ФИПИ

Открытый вариант
КИМ ЕГЭ по информатике 2025

Скачать

https://doc.fipi.ru/ege/otkrytyy-bank-zadaniy-ege/otkrytyye-varianty-kim-ege/2025/Inf_1_ege2025.zip

Задание 26

Задание выполняется с использованием прилагаемых файлов.

В магазине для упаковки подарков есть N кубических коробок. Самой интересной считается упаковка подарка по принципу матрёшки – подарок упаковывается в одну из коробок, та, в свою очередь, в другую коробку и т.д. Одну коробку можно поместить в другую, если длина её стороны хотя бы на 9 единиц меньше длины стороны другой коробки. Определите наибольшее количество коробок, которое можно использовать для упаковки одного подарка, и максимально возможную длину стороны самой маленькой из этих коробок. Размер подарка позволяет поместить его в самую маленькую коробку.

Входные данные

В первой строке входного файла находится число N – количество коробок в магазине (натуральное число, не превышающее 10 000). В следующих N строках находятся значения длин сторон коробок (все числа натуральные, не превышающие 10 000), каждое – в отдельной строке.

Запишите в ответе два целых числа: сначала наибольшее количество коробок, которое можно использовать для упаковки одного подарка, затем максимально возможную длину стороны самой маленькой коробки в таком наборе.

26

В магазине для упаковки подарков есть N кубических коробок. Самой интересной считается упаковка подарка по принципу матрёшки – подарок упаковывается в одну из коробок, та в свою очередь в другую коробку и т.д. Одну коробку можно поместить в другую, если длина её стороны хотя бы на 3 единицы меньше длины стороны другой коробки. Определите наибольшее количество коробок, которое можно использовать для упаковки одного подарка, и максимально возможную длину стороны самой маленькой коробки, где будет находиться подарок. Размер подарка позволяет поместить его в самую маленькую коробку.

Входные данные

В первой строке входного файла находится число N – количество коробок в магазине (натуральное число, не превышающее 10 000). В следующих N строках находятся значения длин сторон коробок (все числа натуральные, не превышающие 10 000), каждое – в отдельной строке.

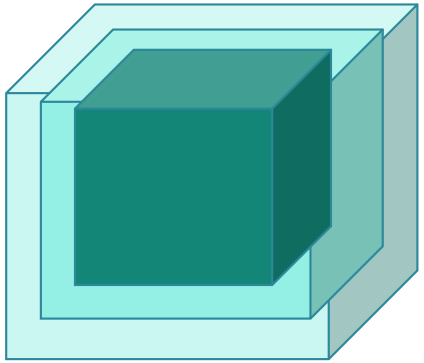
Запишите в ответе два целых числа: сначала наибольшее количество коробок, которое можно использовать для упаковки одного подарка, затем максимально возможную длину стороны самой маленькой коробки в таком наборе.

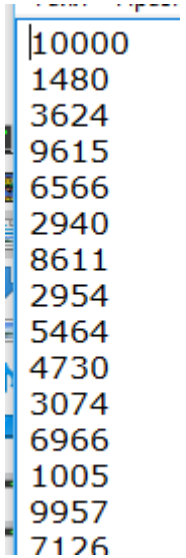
Типовой пример организации данных во входном файле

```
5
43
40
32
40
30
```

Пример входного файла приведён для пяти коробок и случая, когда минимальная допустимая разница между длинами сторон коробок, подходящих для упаковки «матрёшкой», составляет 3 единицы.

При таких исходных данных условию задачи удовлетворяют наборы коробок с длинами сторон 30, 40 и 43 или 32, 40 и 43 соответственно, т.е. количество коробок равно 3, а длина стороны самой маленькой коробки равна 32.





Количество коробок в магазине 10 000.
Одну коробку можно поместить в другую, если длина её стороны хотя бы на 9 единиц меньше длины стороны другой коробки.
Определите наибольшее количество коробок, которое можно использовать для упаковки одного подарка и максимально возможную длину сторон самой маленькой коробки, где будут находиться подарки.
Запишите в ответе два целых числа: сначала наибольшее количество коробок, которое можно использовать для упаковки одного подарка, затем максимально возможную длину стороны самой маленькой коробки в таком наборе.

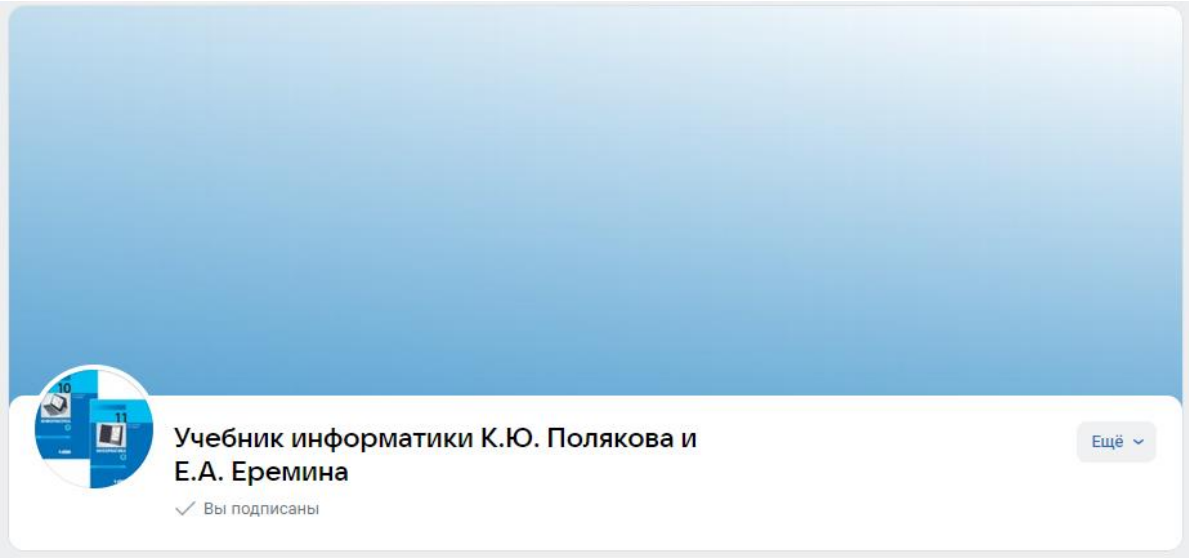
```
1040 57
```

```
File  Edit  Format  Run  Options  Window  Help
with open('1_26.txt') as f:
    n = int(f.readline())
    a = [int(x) for x in f] #формируем список чисел из файла
    a.sort(reverse=True)
    ans = [a[0]]
    for i in range(1,len(a)):
        if ans[-1] - a[i] >= 9:
            ans.append(a[i])
    print(len(ans),ans[-1])
```

Использованы источники:

https://kpolyakov.spb.ru
https://kompege.ru
https://vk.com/inform_web

https://vk.com/kp_probook



Задание

- Решить задачи https://disk.yandex.ru/d/N5uRFinfVuHK3w
- Отправьте файл с решением https://forms.yandex.ru/u/65f4084943f74f93fd3ec3bf/