И Н Ф О Р М А Т И К А

Только самое важное – твоя шпаргалка

ЗАДАНИЕ №2

```
for x in range(2):
  for y in range(2):
    for z in range(2):
      for w in range(2):
        if (выражение из условия) == True (False):
          print(x, y, z, w)
```

В ПРОГЕ	В УСЛОВИЯХ
or	v
and	∧
<=	→
==	≡
not	¬

СПИСКИ

a = [] — создание списка
a.append(n) — добавление элемента n в список
len(a) — длина списка
max(a) — максимальный элемент
min(a) — минимальный элемент
sum(a) — сумма списка
a.sort( ) — сортировка списка в порядке возрастания
a.sort(reverse = True) — сортировка списка в порядке убывания
a.count(n) — количество n в списке

СТРОКИ

a = “” — создание строки
len(a) — длина строки
a.count(“n”) — количество n в строке
“hi” in a — проверка наличия подстроки в строке
a.split(“ab”) — разделение на список строк по ab
“”.join(b) — объединение списка в строку
a = a.replace(“88”, “8”, 1) — замена
“a” < “b” — строки можно сравнивать в алфавитном порядке

ЗАДАНИЕ №5

Программа решения конкретного 5-го задания, реализован вариант с дублированием цифры и добавлением остатка от деления суммы цифр на 2 вместо 97 - пишите нужное вам значение.

```
def task(N):
  n_bin = bin(N)[2:]
  n_bin = n_bin + n_bin[-1]
  n_bin += str(sum([int(i) for i in n_bin]) % 2)
  return int(n_bin,2)
for i in range(1, 1000):
  if task(i) > 97:
    print(task(i)) break
```


ЗАДАНИЕ №6 АВТОПОДСЧЕТ

```
import turtle
t = turtle.Pen( )
t.speed(10000)
k = 100
t.left(90)
t.begin_fill( )
for i in range(3):
  t.forward(111 * k)
  t.right(120)
t.end_fill( )
cnt = 0
canvas = turtle.getcanvas( )
for x in range(-400 * k, 400 * k, k):
  for y in range(-400 * k, 400 * k, k):
    tmp = canvas.find_overlapping(x, y, x, y)
    if tmp == (5, ):
      cnt += 1
print(cnt)
turtle.mainloop( )
```


ЗАДАНИЕ №6 ПРОГА

```
from turtle import *
speed(0)
pendown( )
for _ in range(30):
  forward(4 * 20)
  right(120)
penup( )
for x in range(-10, 10):
  for y in range(-10, 10):
    goto(x * 20, y * 20)
    dot( )
done( )
```

```
from math import *
cnt = 0
for x in range(0, 200):
  for y in range(0, 200):
    if x > 0 and y > tan(radians(30)) * \
x and y < tan(radians(150)) * x + 137:
      cnt += 1
print(cnt)
```


ЗАДАНИЕ №7 ФОРМУЛЫ

V (объем изображения)= N (количество пикселей) * i (глубина цвета = объем пикселя)
Количество цветов в палитре = 2^i (глубина цвета = объем пикселя)
V (размер файла)= N (кол-во волн) * F (частоту) * B (разрешения) * t (время)
Количество уровней дискретизации = 2^B (разрешение)

1 байт = 8 бит
1 Кбайт = 2^{10} байт = 2^{13} бит
1 Мбайт = 2^{10} Кбайт = 2^{20} байт = 2^{23} бит

Когда переводим в биты — умножаем, когда переводим из бит — делим

ЗАДАНИЕ №8

С повторами:

```
from itertools import product
p = product('ШКОЛА', repeat = 3)
n = 0
for x in p:
  if x.count('К') == 1:
    n += 1
print(n)
```

Перестановки, без повторов:

```
from itertools import permutations
p = permutations('ШКОЛА', r=3)
n = 0
for x in p:
  if x.count('К') == 1:
    n += 1
print(n)
```

Системы счисления:

```
from itertools import *
cnt = 1
result = 0
for i in product(sorted("МОНТАЖЕР"), repeat = 6):
  x = "".join(i)
  if cnt % 3 == 0 and x[0] == "О" and 2 <=
x.count("Ж") <= 3:
    result += 1
  cnt += 1
print(result)
```


EXCEL

ЕСЛИ(условие; значение если верно; если неверно)
=ЕСЛИ(A1<A2; A1+1; 0)
ОСТАТ(число; делитель)
=ОСТАТ(L8; 2)
МАКС(число области_1: число области_2)
=МАКС(B2: D8)
МИН(число области_1: число области_2)
=МИН(B2: D8)
СРЗНАЧ(число области1: число области2)
=СРЗНАЧ(B2: D8)
СУММ(число области_1: число области_2)
=СУММ(B2: D8)
НАИБОЛЬШИЙ(число области_1: число области_2; номер)
=НАИБОЛЬШИЙ(B2:D8;3) (вернет 3-й максимальный элемент)

ЗАДАНИЕ №9

```
f = open("9.1.txt")
cnt = 0
for line in f:
  nums = sorted(list(map(int, line.split())))
  repeat, not_repeat = [x for x in nums if nums.count(x) > 1], [x for x in nums
if nums.count(x) == 1]
  if len(repeat) == 2:
    if all(x % 2 != 0 for x in repeat) and all(x % 2 == 0 for x in not_repeat):
      cnt += 1
print(cnt)
```

Условие: одно число повторяется дважды, неповторяющиеся числа чётны, повторяющиеся — нечётны

ЗАДАНИЕ №13

ДЕМОВЕРСИЯ:

```
from ipaddress import *

ip_net = ip_network('167.128.120.83/255.255.255.224',
0)
print(ip_net[-2])
```

КРЫЛОВ:

```
from ipaddress import *
ip_net = ip_network("252.32.33.87/255.255.0.0", False)
cnt = 0
for ip_ad in ip_net:
  if bin(int(ip_ad))[-16:].count("1") > bin(int(ip_ad))[:-16].count("1") * 2:
    cnt += 1
print(cnt)
```


ЗАДАНИЕ №14

```
s = 3 * 5103 ** 2020 + 3 * 729 ** 2021 - 2 * 343 ** 2022 + 27 ** 2023 - 4 * 7 ** 2024 - 2029
cnt = 0
while s > 0:
  if s % 36 > 12: #s % 36 возвращает последнюю цифру числа в 36-ой СС
    cnt += 1
  s //= 36 # s //= 36 удаляет последнюю цифру числа в 36-ой СС
print(cnt)

for x in "0123456789ABCDEFGH":
  s = int(f"AB5{x}3", 18) + int(f"EF{x}13", 18)
  if s % 17 == 0:
    print(s // 17)
```

Вместо "0123456789ABCDEFGH" запишите в строку те символы, которые есть в требуемой системе счисления.
В выражении int(f"AB5{x}3", 18) + int(f"EF{x}13", 18) вместо AB5{x}3 и EF{x}13 запишите нужное по условию выражение, а вместо 18 запишите нужную систему счисления.

ЗАДАНИЕ №14

```
def tr(n):
  s = ""
  alph="0123456789ABC"
  while n > 0:
    s += alph[n % 13]
    n //= 13
  return s[::-1]
for x in range (1, 4600 + 1):
  d = 3*13**210 + 11*13**150 - x
  if tr(d).count('0') ==61:
    print(x)
```


ЗАДАНИЕ №15 ГРАФИКИ И КОНЪЮНКЦИЯ

```
for A in range(1, 1000):
  podoschelA = True
  for x in range(1, 1000):
    for y in range(1, 1000):
      if (ваша функция) == False:
        podoschelA = False
        break
    if podoschelA == False:
      break
  if podoschelA == True:
    print(A)
```

```
for A in range(1, 1000):
  for x in range(0, 1000):
    if (ваша функция) == False:
      break
  else:
    print(A)
```


ЗАДАНИЕ №15 ОТРЕЗКИ

НАИБОЛЬШАЯ ДЛИНА

```
Q = list(range(12, 33))
P = list(range(4, 14))
A = list(range(1, 1000))
for x in range(1, 1000):
  if not(((x in A) <= (x in P)) or (x in Q)):
    A.remove(x)
print(A)
```

НАИМЕНЬШАЯ ДЛИНА

```
Q = list(range(12, 33))
P = list(range(4, 14))
A = []
for x in range(1, 1000):
  if not(((x in A) <= (x in P)) or (x in Q)):
    A.append(x)
print(A)
```

ЗАДАНИЕ №15 ДЕЛ

```
def Del(x, A):
  return x % A == 0
for A in range(1, 1000):
  for x in range(1, 1000):
    if (ваша функция) == False:
      break
  else:
    print(A)
```


ЗАДАНИЕ №16

Одна функция:

```
import sys
sys.setrecursionlimit(10000)
def F(n):
  if n == 1:
    return 1
  if n > 1:
    return (2 * n - 2) * F(n - 1)
print(F(2012) // F(2010))
```

Две функции:

```
def f(n):
  if n == 2:
    return 1
  if n > 2:
    return 4 * f(n - 1) + 2 * g(n - 1) - n * 3 + 8
def g(n):
  if n == 2:
    return 1
  if n > 2:
    return 3 * f(n - 1) + 3 * g(n - 1) + n
print(f(16) + g(16))
```


ЗАДАНИЕ №16 СПИСКИ И КЭШ

```
f = [0] * 2020

for n in range(2020):
 if n == 1:
  f[n] = 1
 if n > 1:
  f[n] = (2 * n - 2) * f[n - 1]

print(f[2012] // f[2010])
```

```
from functools import lru_cache

@lru_cache (None)
def f(n):
  if n < 6:
    return n
  else:
    return f(n - 2) / 3 + 6 * n

for i in range (1, 2500):
  f(i)

print((f(2234) + 3 * f(2232)) //
f(2230))
```


ЗАДАНИЕ №17

```
f = open("17.14.txt")
data = [int(x) for x in f]
res = []
max_13 = max([x for x in data if abs(x) % 100 == 13])
for i in range(len(data) - 2):
  if (int(len(str(abs(data[i]))) == 5) + int(len(str(abs(data[i + 1]))) == 5) + \
int(len(str(abs(data[i + 2]))) == 5)) == 2 and sum(data[i:i + 3]) <= max_13:
    res.append(data[i] + data[i + 1] + data[i + 2])
print(len(res), max(res))
```


ЗАДАНИЕ №18

Случай, когда робот ходит вниз и вправо:

ПОИСК МАКС = ИЗНАЧАЛЬНОЕ ЗНАЧЕНИЕ + МАКС(ЗНАЧЕНИЕ ИЗ ТАБЛИЦЫ СУММ СПРАВА, ЗНАЧЕНИЕ ИЗ ТАБЛИЦЫ СУММ СНИЗУ)

ПОИСК МИН = ИЗНАЧАЛЬНОЕ ЗНАЧЕНИЕ + МИН(ЗНАЧЕНИЕ ИЗ ТАБЛИЦЫ СУММ СПРАВА, ЗНАЧЕНИЕ ИЗ ТАБЛИЦЫ СУММ СНИЗУ)

Удаляйте варианты, которые ведут в стенки.

ЗАДАНИЯ №19-21

Две кучи:

```
def g(x, s, p, end):
 if (x + s) >= 156 and p in end:
  return True
 if (x + s) < 156 and p == max(end):
  return False
 if (x + s) >= 156 and p not in end:
  return False
 if (p + 1) % 2 == (end[0] % 2):    return (g(x + 2, s, p + 1, end) or g(x * 5, s, p + 1, end) or g(x, s + 2, p + 1, end) or g(x, s * 5,
p + 1, end))
 else:
  return (g(x + 2, s, p + 1, end) and g(x * 5, s, p + 1, end) and
g(x, s + 2, p + 1, end) and g(x, s * 5, p + 1, end))

task19, task20, task21 = [], [], []
x = 5
for s in range(1, 151):
 if g(x, s, 0, [1]):
  task19.append(s)
 if g(x, s, 0, [3]):
  task20.append(s)
 if g(x, s, 0, [2, 4]) and not(g(x, s, 0, [2])):
  task21.append(s)
print(task19) # Ответ: 31
print(task20) # Ответ: 28
print(task21) # Ответ: 29
```


ЗАДАНИЯ №19-21

Короткий код:

```
def g(s, p, end):
 if s >= 100: return p in end
 if s < 100 and p == max(end): return False
 moves = [g(s + 1, p + 1, end), g(s * 2, p + 1, end)]
 return any(moves) if (p + 1) % 2 == (end[0] % 2) else all(moves)

print([s for s in range(1, 100) if g(s, 0, [1])]) # Ответ: 50
print([s for s in range(1, 100) if g(s, 0, [3])]) # Ответ: 48
print([s for s in range(1, 100) if g(s, 0, [2, 4])]) # Ответ: 47
```


ЗАДАНИЯ №19-21

Одна куча:

```
def g(s, p, end):
 if s >= 100 and p in end:
  return True
 if s < 100 and p == max(end):
  return False
 if s >= 100 and p not in end:
  return False
 if (p + 1) % 2 == (end[0] % 2):
  return g(s + 1, p + 1, end) or g(s * 2, p + 1, end)
 else:
  return g(s + 1, p + 1, end) and g(s * 2, p + 1, end)
task19, task20, task21 = [], [], []
for s in range(1, 100):
 if g(s, 0, [1]):
  task19.append(s)
 if g(s, 0, [3]):
  task20.append(s)
 if g(s, 0, [2, 4]) and not(g(s, 0, [2])):
  task21.append(s)
print(task19) # Ответ: 50
print(task20) # Ответ: 48
print(task21) # Ответ: 47
```

ЗАДАНИЕ №23

Команды +2, *2, исключая 6:

```
def rec(start, end):
  if start > end or start == 6: return 0
  if start == end: return 1
  return rec(start + 2,end) + rec(start * 2,end)
print(rec(2, 20))
```

Команды -2, -5:

```
def rec(start, end):
  if start < end: return 0
  if start == end: return 1
  return rec(start - 2,end) + rec(start - 5,end)
print(rec(23, 2))
```


ЗАДАНИЕ №24

Сколько пар идущих подряд AB или AC?

```
f = open("24.txt")
s = f.readline()
s = s.replace("AC", "AB")
for i in range(1, 1000):
  ss = "AB" * i
  if ss in s:
    print(i)
```

Наибольшая строка без YZXZ.

```
f = open("24.txt")
s = f.readline()
a = s.split("YZXZ")
print(len(max(a, key=len)) + 6)
print(a.index(max(a, key = len))) #не забудьте проверить индекс строки
```


ЗАДАНИЕ №24

Наибольшая строка, где ни одна цифра не стоит рядом с цифрой, а буква — рядом с буквой.

```
f = open("24.txt")
s = f.readline()
s = s.replace("A", "B").replace("B", "C").replace("CC", "C C").replace("CC", "C C")
s = s.replace("8", "9").replace("99", "9 9").replace("99", "9 9")
s = s.split()
print(len(max(s, key = len)))
```

Наименьшая строка, где есть 310 букв K.

```
f = open("24.2.txt")
s = f.readline()
index_K = []
for i in range(len(s)-1):
  if s[i] == "K":
    index_K.append(i)
min_len = 10**8
for i in range(309, len(index_K)):
  min_len = min(min_len, index_K[i] - index_K[i - 309] + 1)
print(min_len)
```


ЗАДАНИЕ №25

Пример задачи на маску 123*5 до 1000000, кратную 42.

```
from fnmatch import *
for x in range(42, 10 ** 6 + 1, 42):
  if fnmatch(str(x), r"123*5"):
    print(x, x // 34)
```

Пример задачи на маску 12345?7?8 до 10^9, кратную 42.

```
from fnmatch import *
for x in range(42, 10 ** 9 + 1, 42):
  if fnmatch(str(x), r"12345?7?8"):
    print(x, x // 34)
```

Пример задачи на маску 12345*7? до 10^9, кратную 42.

```
from fnmatch import *
for x in range(42, 10 ** 9 + 1, 42):
  if fnmatch(str(x), r"12345*7?"):
    print(x, x // 34)
```


ЗАДАНИЕ №24

ЕГЭ 2025

```
import re
f = open("24_5.txt")
s = f.read()
pattern = r'[02468]([A-Z])\1+[02468]'
ans = 0
for match in re.finditer(pattern, s):
  ans = max(ans, len(match.group()))
print(ans)
```

ЕГЭ 2025

```
f = open("24_3hm.txt")
s = f.read()
s = ''.join('#' if c in '02468' else c for c in s)
ans = 0
for t in s.split('#')[1:]:
  p = [i for i, c in enumerate(t) if c == 'F']
  if len(p) >= 58:
    ans = max(ans, len(t) + 1 if len(p) == 58 else p[58] + 1)
print(ans)
```

ДОСРОК 2025

```
import re

f = open('file.txt')
s = f.readline()
valid_list = re.findall('[123456789AB][0123456789AB]*[02468A]', s)
print(len(max(valid_list, key = len)))
```

ЕГЭ 2025

```
f = open('24_3.txt')
s = f.read()
ans = 0
for t in s.split('H')[:-1]:
  p = [i for i, c in enumerate(t) if c in '13579']
  if len(p) >= 34:
    ans = max(ans, len(t) + 1 if len(p) == 34 else len(t) - p[-35])
print(ans)
```


ЗАДАНИЕ №24

Арифметика Основа 2024

```
import re
f = open("24.2.txt")
s = f.readline()
is_valid = re.findall(r'(?:[234][0234]*)(?:[+*](?:[234][0234]*))*', s)
print(len(max(is_valid, key=len)))
```

```
f = open("24.2.txt")
s = f.readline()
s = s.replace("+", "*")
a = s.split("*")
arifm = ""
max_arifm = ""
for x in a:
  if len(x) != 0 and x[0] != "0":
    arifm += x + "*"
  elif len(x) > 0 and x[0] == "0" and int(x) != 0:
    arifm = str(int(x)) + "*"
  else:
    arifm = ""
  max_arifm = max(arifm, max_arifm, key=len)
print(len(max_arifm) - 1)
```


ЗАДАНИЕ №24

Арифметика Резерв 2024

```
f = open("24.4dz.txt")
s = f.readline()
s = s.replace("++", "+ +").replace("++", "+ +")
s = s.replace("**", "* *").replace("**", "* *")
s = s.replace("+*", "+ *").replace("+*", "+ *")
s = s.replace("*+", "* +").replace("*+", "* +")
a = s.split()
max_buf = ""
for x in a:
  x2 = x[:]
  if x[0] in "+*":
    x2 = x2[1:]
  if x[-1] in "+*":
    x2 = x2[:-1]
  x3 = x2.split("+")
  buf = ""
  for pod_summi in x3:
    if len(pod_summi) > 0 and eval(pod_summi) == 0:
      buf += pod_summi + "+"
    elif len(pod_summi) > 0:
      if "0*" in pod_summi:
        buf = pod_summi[pod_summi.index("0*"):] + "+"
      elif pod_summi.endswith("0"):
        buf = "0*"
      else:
        buf = ""
    else:
      buf = ""
    max_buf = max(buf, max_buf, key=len)
print(len(max_buf) - 1)
```

И Н Ф О Р М А Т И К А

Только самое важное – твоя шпаргалка

ЗАДАНИЕ №26 ГРУЗЫ

```
f = open("26.txt")
s, n = map(int, f.readline().split())
a = []
for i in range(n):
  a.append(int(f.readline()))
a.sort()
b = []
for i in range(n):
  if sum(b) + a[i] <= s:
    b.append(a[i])
  elif sum(b) - b[-1] + a[i] <= s:
    del b[-1]
    b.append(a[i])
  else:
    break
print(len(b), max(b))
```


ЗАДАНИЕ №26 ТЕАТР

Вычитать или добавлять должны на +1 от количества свободных мест.

```
f = open("26.txt")
n = int(f.readline())
a = []
for i in range(n):
  a.append(list(map(int, f.readline().split())))
a.sort()
b = []
for i in range(n-1):
  if (a[i][0] == a[i + 1][0]) and (a[i][1] + 18 == a[i + 1][1]):
    b.append([a[i][0], a[i][1]+1])
print(b)
```


ЗАДАНИЕ №26 КОРОБКИ

```
f = open("26.1.txt")
n = int(f.readline())
s = sorted([int(x) for x in f], reverse = true)
korobki = [s[0]]
for i in range(1, n):
  if korobki[-1] >= s[i] + 5:
    korobki.append(s[i])
print(len(korobki), min(korobki))
```


ЗАДАНИЕ №26 КОРОБКИ И ЛЕНТЫ

```
f = open("26.10.1.txt")
n, m = map(int, f.readline().split())
korobki = []
lents = []
for i in range(m):
  a, b = map(int, f.readline().split())
  korobki.append(a)
  lents.append(b)
for i in range(n - m):
  a = int(f.readline())
  korobki.append(a)
korobki.sort(reverse=True)
kuda_kladem = []
for x in korobki:
  if kuda_kladem == [] and x in lents:
    kuda_kladem.append(x)
  elif kuda_kladem == [] and x not in lents:
    continue
  elif x <= kuda_kladem[-1] - 7 and x in lents:
    kuda_kladem.append(x)
print(len(kuda_kladem), max(kuda_kladem) - min(kuda_kladem))
```


ЗАДАНИЕ №26 СКИДКИ

```
f = open("26.web_6.txt")
N = int(f.readline())
discounts = []
summ = 0
max_value = 0
for i in range(N):
  x = int(f.readline())
  if x <= 80:
    summ += x
  else:
    discounts.append(x)
discounts.sort()
for i in range(len(discounts)):
  if i < (len(discounts) // 2):
    summ += (discounts[i] * 0.55)
    max_value = discounts[i]
  else:
    summ += discounts[i]
print(int(summ) + 1, max_value)
```


ЗАДАНИЯ №26 С АПРОБАЦИИ

```
f = open("26.txt")
n = int(f.readline())
pokupki = []
for s in f:
  pokupki.append(int(s))
pokupki.sort(reverse = True)
print(sum(pokupki[n // 3:]))
summ = sum(pokupki)
for i in range(2, len(pokupki),
3):
  summ -= pokupki[i]
print(summ)
```


ЗАДАНИЕ №26 ЗАЛЫ

```
f = open("26.txt")
n = int(f.readline())
data = []
for s in f:
  start, end = map(int, s.split())
  data.append([end, start])
data.sort()
events = []
for event in data:
  if events == []:
    events.append(event)
  elif events[-1][0] <= event[1]:
    events.append(event)

for event in data:
  if event[1] >= events[-2][0] and event[1] >= events[-1][1]:
    del events[-1]
    events.append(event)

print(len(events), events[-1])
```

ЗАДАНИЕ №26 РЕЗЕРВ

```
f = open("26_1.txt")
n = int(f.readline())
process = [0] * 1442

for _ in range(n):
  start, end = map(int, f.readline().split())
  process[start] += 1
  process[end + 1] -= 1
process = [x for x in process if x != 0]

current_process = max_process = max_piks = 0
for minute in range(len(process)):
  current_process += process[minute]

  if current_process > max_process:
    max_process = current_process
    max_piks = 1
  elif current_process == max_process:
    max_piks += 1

print(max_piks, max_process)
```


ЗАДАНИЕ №26 ШЛИФОВКА

```
f = open("26.2.txt")
n = int(f.readline())
start_lenta = []
end_lenta = []
for i in range(n):
  shlif, okr = map(int, f.readline().split())
  if shlif < okr:
    start_lenta.append([shlif, i + 1])
  else:
    end_lenta.append([okr, i + 1])
print(max(end_lenta)[-1], len(start_lenta))
```


ЗАДАНИЕ №26 БАГАЖИ

```
f = open("26.3.txt")
n, k = map(int, f.readline().split())
data = sorted([list(map(int, x.split())) for x in f])
count_file, last_disk = 0, 0
cells = [0] * k

for i in range(n):
  start, end = data[i]
  for j in range(k):
    if cells[j] == 0:
      cells[j] = [start, end]
      last_disk = j + 1
      count_file += 1
      break
    elif cells[j][1] + 1 <= start:
      cells[j] = [start, end]
      last_disk = j + 1
      count_file += 1
      break
print(count_file, last_disk)
```


ЗАДАНИЕ №27

```
f = open('task27_1/27A.txt')
f.readline()
roots = [list(map(float, s.replace(",", ".").split())) for s in f]
clusters = [[], []]

for x, y in roots:
  if y < -5:
    clusters[0].append((x, y))
  else:
    clusters[1].append((x, y))

best_centroids = [[] for i in range(len(clusters))]
for i in range(len(clusters)):
  min_dist = 10**10
  for x1, y1 in clusters[i]:
    dist = 0
    for x2, y2 in clusters[i]:
      euclidian_dist = ((x2 - x1)**2 + (y2 - y1)**2)**0.5
      dist += euclidian_dist
    if dist < min_dist:
      min_dist = dist
      best_centroids[i] = [x1, y1]

P_x = sum([x / len(clusters) for x, y in best_centroids]) * 10_000
P_y = sum([y / len(clusters) for x, y in best_centroids]) * 10_000
print(int(P_x), int(P_y))
```


ЗАДАНИЕ №27A

```
f = open('task27_1/27A.txt')
f.readline()
roots = [list(map(float, s.replace(",", ".").split())) for s in f]
mn = 10**10
ans = []
for i in range(len(roots) - 1):
  for j in range(i + 1, len(roots)):
    a = roots[i]
    b = roots[j]
    s = 0
    for d in roots:
      d1 = ((d[0] - a[0]) ** 2 + (d[1] - a[1]) ** 2) ** 0.5
      d2 = ((d[0] - b[0]) ** 2 + (d[1] - b[1]) ** 2) ** 0.5
      s += min(d1, d2)
    else:
     if s < mn:
      mn = s
      ans = [a, b]
print(int((ans[0][0] + ans[1][0]) / 2 * 10000), \
   int((ans[0][1] + ans[1][1]) / 2 * 10000))
```


ЗАДАНИЕ №27 DBSCAN

```
import math
from turtle import *
from math import dist
def visual(clusters):
  k = 40
  penup(), tracer(0)
  colors = ['green', 'blue', 'brown', 'red', 'yellow', 'pink', 'black']
  for i in range(len(clusters)):
    for x, y in clusters[i]:
      setpos(x*k, y*k)
      dot(5, colors[i])
  done()

def centroid(cluster):
  c = []
  for x1, y1 in cluster:
    c += [[sum(math.dist((x1, y1), (x2, y2)) for x2, y2 in cluster), (x1, y1)]]
  return min(c)[1]

f = open('27dbscan/27_2/27A.txt')
f.readline()
points = [list(map(float, s.replace(",", ".").split())) for s in f]
clusters, epsilon = [], 0.25
while points:
  clusters.append([points[0]])
  del points[0]
  for p1 in clusters[-1]:
    for p2 in points[:]:
      if math.dist(p1, p2) < epsilon:
        clusters[-1].append(p2)
        points.remove(p2)
  if len(clusters[-1]) <= 10:
    del clusters[-1]

best_centroids = [[] for i in range(len(clusters))]
for i in range(len(clusters)):
  min_dist = 10**10
  for x1, y1 in clusters[i]:
    dist = 0
    for x2, y2 in clusters[i]:
      euclidian_dist = math.dist([x1, y1], [x2, y2])
      dist += euclidian_dist
    if dist < min_dist:
      min_dist = dist
      best_centroids[i] = [x1, y1]
print(best_centroids)
visual(clusters)
P_x = sum([x for x, y in best_centroids]) / len(clusters)
P_y = sum([y for x, y in best_centroids]) / len(clusters)
P_s = sum([len(cluster) / (4 * 4) for cluster in clusters])
print(int((P_x + P_y) * 10000), int(P_s * 1000))
```