Шпаргалка. Прога КЕГЭ. ШКОЛКОВО.

Задание №2

```
# 1 СПОСОБ
print('x y z w f')
for x in range(2):
  for y in range(2):
    for z in range(2):
      for w in range(2):
        if (логическая функция == 0 или 1):
          print(x, y, z, w)
# 1) Получаем набор x, y, z, w для которых
# функция ложна или истинна
# в зависимости от условия задачи
# 2) Аккуратно сопоставляем с таблицей
# в условии
```

```
# 2 СПОСОБ
from itertools import product
print('x y z w')
for x, y, z, w in product((0, 1), repeat=4):
# repeat - кол-во переменных
  if логическая функция == 0 или 1:
    print(x, y, z, w)
```

Задание №8

Прототип: "Сколько 4-буквенных слов сможет составить N из букв А, Б, В, Г, если при этом А не должна стоять на первом месте и должно быть ровно две буквы Б в слове?

ВЛОЖЕННЫМИ ЦИКЛАМИ

```
letters = 'АБВГ' #сюда записываем буквы
s = set()  #создаем множество, дабы не
           запутаться со счетчиком
for q in letters:
  for w in letters:
    for e in letters:
      for r in letters: #создаем столько
                        циклов, сколько
                        букв в слове
        word = q+w+e+r
        if (word[0] != 'А' and
          word.count('Б') == 2):
        #прописываем все условия
          s.add(word)
print(len(s))
```

ЧЕРЕЗ ITERTOOLS

```
from itertools import product
ans = 0
comb = set(product('АБВГ', repeat = 4))
for i in comb:
  s = ''.join(i)
  if s[0] != 'А' and s.count('Б') == 2:
    ans += 1
print(ans)
```

Прототип: "Все 4-буквенные слова, составленные из букв Б, Ш, А, В, записаны в алфавитном порядке. Вот начало списка:
1. АААА
2. ААБ
3. ААШ
4. ААВ
5. ААБА
Укажите номер слова "БББ"

```
letters = 'АБШВ' #записываем буквы в
                 том порядке, в каком
                 они стоят в нумерации
k = 0  #создаем счетчик с нумерацией
       с нуля
for q in letters:
  for w in letters:
    for e in letters:
      for r in letters: #создаем столько
                        циклов, сколько
                        букв в слове
        word = q+w+e+r
        k += 1 #добаляем в счетчик +1,
               когда записываем
               новое слово
        if word == 'БББ':
          print(k)
```

Задание №6

```
# Запускаем перебор значений,
#которое вводится в программу
for t in range(-1000, 1000):
# промежуток перебора можеть
#быть другим
  x = t # если в коде вводится
        переменная x, то
        присваиваем её t
  # сюда вставляем код программы,
    где вводится переменная x
  # ...
  if (n == 60): # если по коду
  выводится n и по условию
  вывелось 60
    print(t)
```

Задание №14

```
s = *сюда выражение*
x = int(s)
r = *номер СС до 10*
result = ''
while x > 0:
  result = str(x % r) + result
  x //= r

print(result.count('1')) # количество
                         символов '1'
result = list(result) # находим сумму и
                      произведение всех
                      цифр в числе
summ = 0
mul = 1
for elem in result:
  summ += int(elem)
  mul *= int(elem)
print(summ, mul)

# Универсальная функция перевода
в любые СС
def convert_base(num, to_base=10,
from_base=10):
  if isinstance(num, str):  # проверяем,
                            является ли
                            num строкой
    n = int(num, from_base)
  else:
    n = int(num)
  alphabet = "0123456789ABCDEFGHI
              JKLMNOPQRSTUVWXYZ"
  if n < to_base:
    return alphabet[n]
  return convert_base(n // to_base,
   to_base) + alphabet[n % to_base]

#Встроенная функция перевода из
любой СС в десятичную
n = int(a, 3) #перевод из троичной
              в десятичную.
              Важное условие: a
              должно быть строкой
```

Задание №12

ИСПОЛНИТЕЛЬ РЕДАКТОР

```
s = *заданное выражение*
while s.find(s1) != -1:
# while - пока
# s.find(s1) - находит индекс первого
вхождения строки s1 в s
# если не нашёл - возвращает -1
# вместо s1 можно писать нужные символы
в кавычках, как ниже
while s.find('5') != -1:
  s = s.replace('5', '77', 1)
# s.replace(s1, s2, 1) заменяет первое
вхождение s1 в s на s2
print(sum([int(s[i]) for i in range(len(s))]))
# строка выше печатает сумму цифр
полученного числа
```

ИСПОЛНИТЕЛЬ ЧЕРТЕЖНИК

Условие: чертёжнику был дан для исполнения следующий алгоритм:
НАЧАЛО
  сместиться на (-30, -110)
  ПОВТОРИ N РАЗ
    сместиться на (a,b)
    сместиться на (76,–93)
  КОНЕЦ ПОВТОРИ
  сместиться на (0,5)
КОНЕЦ

Вопрос: чему равно максимальное значение N при котором найдутся такие значения чисел a и b что после выполнения программы Чертёжник возвратится в исходную точку?

Решение
Чертёжник остановился в точке (0, 0), значит конечные координаты равны этим числам.

x = -30 + N · (a + 76) = 0

y = -110 + N · (b - 93) + 5 = 0

Перенесём все числа в левые части:

N · (a + 76) = 30 = 2 · 3 · 5

N · (b - 93) = 105 = 3 · 5 · 7

Для выбранных N должны подобраться целые значения в скобках. **Значит N должно быть делителем чисел 30 и 105**. Максимальное такое число - 15.

Задание №15

НЕРАВЕНСТВА

```
def f(x, y, A):
  return *функция из условия*

for A in range(1, 200):
  flag = True # предполагаем, что с этим
              A всё хорошо
  for x in range(1, 200):
    for y in range(1, 200):
      if not f(x, y, A): # в условии нужно
                         истинно, поэтому
                         если
      # при этом A - ложно, выходим
      с флагом False
        flag = False
        break
    if not flag:
      break
  if flag:
    ans = A # максимальный ответ
            будет присвоен на
            последней операции
print(ans)
```

ПОРАЗРЯДНАЯ КОНЪЮНКЦИЯ

```
def f(x, a):
  return *функция*


for a in range(0, 300):
  p = True # предполагаем всё хорошо
  for x in range(0, 300):
    if f(x, a) == False: # если не хорошо -
                         противоречит условию
                         задачи:
    # выходим флагом False
      p = False
      break
  if p == True:
    print(a)
```

Задание №15 (продолжение)

ЧИСЛОВОЙ ОТРЕЗОК

#1 СПОСОБ

```
def inn(x, A):
    return A[0] <= x <= A[1] # функция принадлежности числа к отрезку
    # предполагаем, что A - это массив,
    # где 0-ой элемент - начало отрезка
    # а 1-ый элемент - конец отрезка

def f(x, A):
    P = [a1, a2]
    Q = [a2, b2]
    K = [a3, b3]
    return *функция*

borders = [0, 0]
minim = 100000
k = 5 # воспользуемся более точным приближением при подборе
for a in range(0, 80 * k):
    for b in range(a, 80 * k):
        A = [a / k, b / k]
        good = True # предполагаем, что при этом A всё хорошо
        for x in range(0, 100 * k):
            if not f(x/k, A): # оказалось не хорошо, по условию нужно истинно,
                # а получили ложь, выходим с флагом False
                good = False
                break
        if good: # всё хорошо, отрезок такой походит
            if A[1] - A[0] <= minim:
                minim = A[1] - A[0] # сохраняем длину
                borders = A.copy() # сохраним сам отрезок
print(minim)
print(borders)
```

#2 СПОСОБ

```
# Задаем отрезки
p = range(10, 29 + 1)
q = range(13, 18 + 1)

# Для максимального отрезка: создаем множество со значениями от 1 до 1000
a = set(range(1, 1000))
# Для минимального отрезка: создаем пустое множество
a = set()

for x in range(1, 1000):
    if логическая функция == 0 или 1:
        # Для максимального отрезка: выкидываем такие иксы из отрезка
        a.remove(x)
        # Для минимального отрезка: добавляем такие иксы в отрезок
        a.add(x)

# Количество элементов в отрезке
print(sorted(a))
# Максимальная длина отрезка
print(sorted(a)[-1] - sorted(a)[0])

# 3 способ (в действительных числах)
# Задаем отрезки
p = [i / 10 for i in range(50, 300 + 1)]
q = [i / 10 for i in range(140, 240 + 1)]

# Для максимального отрезка: создаем множество со значениями от 1 до 1000
a = set([i / 10 for i in range(10000)])
# Для минимального отрезка: создаем пустое множество
a = set()

for x in [i / 10 for i in range(10000)]:
    if логическая функция == 0 или 1:
        # Для максимального отрезка: выкидываем такие иксы из отрезка
        a.remove(x)
        # Для минимального отрезка: добавляем такие иксы в отрезок
        a.add(x)

# Количество элементов в отрезке
print(sorted(a))
# Максимальная длина отрезка
print(sorted(a)[-1] - sorted(a)[0])
```

Задание №19-21

ОДНА КУЧА

```
from functools import lru_cache
def moves(heap):
    return heap + 1, heap * 2

@lru_cache(None)
def game(heap):
    if heap >= 31:
        return 0
    steps = [game(x) for x in moves(heap)]
    # Если хотя бы один ход приводит нас к победе
    if any(x % 2 == 0 for x in steps):
        # Возвращаем этот ход
        return 1 + min(x for x in steps if x % 2 == 0)
    # Максимально оттягиваем свое поражение
    return max(steps) + 1

for s in range(1, 31):
    print(s, game(s), [game(x) for x in moves(s)])
```

ДВЕ КУЧИ

```
from functools import lru_cache
def moves(heap):
    a, b = heap
    return (a + 1, b), (a, b + 1), (a * 2, b), (a, b * 2)

@lru_cache(None)
def game(heap):
    if sum(heap) >= 47:
        return 0
    steps = [game(x) for x in moves(heap)]
    # Если хотя бы один ход приводит нас к победе
    if any(x % 2 == 0 for x in steps):
        # Возвращаем этот ход
        return 1 + min(x for x in steps if x % 2 == 0)
    # Максимально оттягиваем свое #поражение
    return max(steps) + 1

for s in range(1, 43):
    print(s, game((4, s)), [game(x) for x in moves((4, s))])
```

УБЫВАНИЕ КАМНЕЙ

```
from functools import lru_cache
def moves(heap):
    # Создаем массив, в который будет добавлять ходы которые можно совершить
    m = []
    if heap >= 1:
        m += [heap - 1]
    if heap >= 3:
        m += [heap - 3]
    return m

@lru_cache(None)
def game(heap):
    if heap <= 7:
        return 0
    steps = [game(x) for x in moves(heap)]
    # Если хотя бы один ход приводит нас к победе
    if any(x % 2 == 0 for x in steps):
        # Возвращаем этот ход
        return 1 + min(x for x in steps if x % 2 == 0)
    # Максимально оттягиваем свое поражение
    return max(steps) + 1


for s in range(100, 7, -1):
    print(s, game(s), [game(x) for x in moves(s)])
```

Задание №22

```
# Запускаем перебор значений, которое вводится в программу
for t in range(-1000, 1000): # промежуток перебора можеть быть другим
    x = t # если в коде вводится переменная x, то присваиваем её t
    # сюда вставляем код программы, где вводится переменная x
    # ...
    if (n == 60 and k == 70): # если по коду выводится n и k, и по условию вывелось 60 и 70
        print(t)
```

Задание №23

СТАНДАРТНЫЙ ПРОТОТИП

“Исполнитель N преобразует число на экране. У исполнителя есть две команды:
1. Умножить на 3
2. Прибавить 1
Сколько существуют программ, для которых при числе 1 рез-том будет 28, при этом траектория вычислений содержит число 5 и не содержит число 17?”

```
a = [0] * 84 #создаем массив на все ходы
a[1] = 1
for i in range(1, 5): #идем до "связующего узла", через который должны пройти все прогр.
    a[i + 1] += a[i]
    a[i * 3] += a[i]
for i in range(6, 84): #обнуляем все остатки путей, которые прошли не через 5
    a[i] = 0
for i in range(5, 28):  #продолжаем от "связующего узла"
    if i == 17:    #проверяем, что число не равно 17
        a[i] = 0
    a[i + 1] += a[i]
    a[i * 3] += a[i]
print(a[28])
```

ИДЕЯ ДИНАМИКИ

Задание №23 (прод.)

“Исполнитель N преобразует число на экране. У исполнителя есть две команды:
1. Умножить на 3
2. Прибавить 1
Сколько существуют программ, для которых при числе 1 рез-том будет 28, при этом траектория вычислений содержит число 5 и не содержит число 17?”

```
def f(st, fn, flag, flag_number, exit_number):
    if st == fn and flag:
        return 1
    if st > fn or st == exit_number:
        return 0
    if st == flag_number: # поднимаем флаг, если дошли до требуемого значения
        flag = True
    x = f(st + 1, fn, flag, flag_number, exit_number)
    y = f(st * 3, fn, flag, flag_number, exit_number)
    return x + y
print(f(1, 28, False, 5, 17))
```

ИДЕЯ РЕКУРСИИ

“СОДЕРЖИТ СТОЛЬКО-ТО КОМАНД”

“Исполнитель Щелчок преобразует число на экране. У исполнителя есть три команды:
1. Прибавить 3
2. Прибавить 1
3. Прибавить само число
Сколько существует программ, для которых при исходном числе 1 результатом является число 28, при этом программа содержит 9 команд, а траектория обязательно не проходит через число 11 и проходит через 13?”

ДИНАМИКА

```
a = [[0] * 10 for i in range(100)]  #создаем двумерный массив: в первой ячейке будем
a[1][0] = 1                         #хранить программы, во второй - команды

for i in range(1, 13):
    for j in range(9):
        if i == 11:
            a[i][j] = 0
        a[i + 3][j + 1] += a[i][j]
        a[i + 1][j + 1] += a[i][j]
        a[i + 1][j + 1] += a[i][j]
for i in range(14, 100):
    for j in range(9):
        a[i][j] = 0
for i in range(13, 28):
    for j in range(9):
        a[i + 3][j + 1] += a[i][j]
        a[i + 1][j + 1] += a[i][j]
        a[i + 1][j + 1] += a[i][j]
print(a[28][9])
```

РЕКУРСИЯ

```
def f(st, fn, flag, comm):
    if st > fn:
        return 0
    if st == 11:
        return 0
    if st == 13:
        flag = True
    if comm < 0:
        return 0
    if st == fn and flag and comm == 0:
        return 1
    return f(st+3,fn,flag,comm-1)+
+f(st+1,fn,flag,comm-1)+f(st*2,fn,flag,comm-1)
print(f(1, 28, False, 9))
```

Задание №24

```
f = open('24.txt') # откроем файл
# Если задача на одну строку
s = f.readline() # считаем одну строку
# Если задача на несколько строк, то проще их поместить
a = f.readlines() # считываем несколько строк и кладём их в массив

len(s) # длина строки s
s.count('AR') # число подстрок 'AR' внутри строки s
s.find('AR') # поиск подстроки 'AR' слева, вычисляет индекс первого вхождения, иначе -1
s.rfind('AR') # поиск подстроки 'AR' справа, вычисляет индекс последнего вхождения, иначе -1
s = s.replace('AR', 'MO', 1) # замена подстроки 'AR' на подстроку 'MO' одно первое вхождение слева

# прототип поиска максимальной длинной цепочки одинаковых символов
s = f.readline()
cur_len = 1
max_len = 0
for i in range(len(s)-1):
    if s[i] == s[i + 1]:
        cur_len += 1
        max_len = max(max_len, cur_len)
    else:
        cur_len = 1
print(max_len)
```

Не забывайте, что иногда можно идти нестандартными путями в задаче, например, воспользоваться помощью .split() и считать разбитые подцепочки.

Файл состоит из символов A, B, C, D, E. Найдите количество символов между двумя самыми отдалёнными буквами A, между которыми находятся только согласные буквы.

```
n = open('file.txt').readline()
a = n.split('A')
maxim = 0
Flag = True
for i in a:    #идем по массиву
    for x in i:  #теперь по строке
        if x not in 'BCD':
            Flag = False
    if Flag:
        maxim = max(maxim, len(i))
    Flag = True
print(maxim)
```

Важно помнить, что если вы используете .split() для задачи, где нужно подсчитать максимальную длину, не содержащую что-то, рассмотрите боковые символы от разбитой строчки.

НАЙТИ И ВЫВЕСТИ САМЫЙ ЧАСТО ПОВТОРЯЮЩИЙСЯ СИМВОЛ

Пример: “Найдите самый часто повторяющийся символ между символами A и R (в данном порядке). Если таких символов несколько, следует взять тот, что раньше по алфавиту.”

ЧЕРЕЗ МАССИВ С АЛФАВИТОМ

```
n = open("file.txt").readline()
letters = 'ABCDEFGHIJKLMNOP
          QRSTUVWXYZ'
array = [0] * 26
for i in range(1, len(n)-1):
    if n[i-1] == 'A' and n[i+1] == 'R':
        array[letters.index(n[i])] += 1
for i in range(len(array)):
    if array[i] == max(array):
        print(letters[i])
        break
```

ЧЕРЕЗ ORD, CHR

```
n = open("file.txt").readline()
array = [0] * 200
for i in range(1, len(n)-1):
    if n[i-1] == 'A' and n[i+1] == 'R':
        array[ord(n[i])] += 1
print(array)
for i in range(len(array)):
    if array[i] == max(array):
        print(chr(i))
        break
```

24 ПРОДОЛЖЕНИЕ

Задание №25

```
# Функция нахождения
количества делителей числа
def count_div(n):  # Счетчик всех
                     делителей
    k = 2  # 1 и само число -
             делители числа всегда
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            k += 1
            if n // i != i:
                k += 1
    return k

# Функция нахождения
отсортированного массива
нетривиальных делителей
def dividers(n):
    a = []
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            a.append(i)
            if n // i != i:
                a.append(n// i)
    a.sort()
    return a

# Функция проверки числа
на простоту
def is_prime(n):
    if n == 1: return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

# Функция нахождения
минимального делителя (после 1)
def min_divider(n):
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return i
```

Задание №27

ПОСЛЕДОВАТЕЛЬНОСТИ (ПРЕФ. СУММЫ)

Найти цепочку с макс. суммой, кратной 42

```
f = open('27.txt')
n = int(f.readline())
s = 0
ans = -1000000
mps = [10000000] * 42 #минимальные преф.
                        суммы элементов,
                        кратных всем числам
                        до 42 вкл.
mps[0] = 0  #т.к. мин. преф. сумма,
            кратная 42, является нулем
for i in range(n):
    s += int(f.readline())    #считаем текущ. сумму
    ans = max(ans, s-mps[s % 42])   #динамически
                                    перезаписываем ответ
    mps[s % 42] = min(mps[s % 42], s)  #кладем
                                       минимальную преф.
                                       сумму в нужную ячейку
print(ans)
```

ДВИЖЕНИЕ ПО ОКРУЖНОСТИ

```
f = open('27.txt')
n = int(f.readline())
a = [int(i) for i in f]
s = [0] * n
current_sum = 0     #задаем первую полуокружность
for i in range(n//2):
    current_sum += a[i]
s[0] = current_sum
for i in range(1, n):    #задаем следующие
                         полуокружности
    s[i] = s[i - 1] - a[i - 1] + a[(i - 1 + n // 2) % n]

cost = 0
for i in range(n):
    cost += (min(i, n - i) * a[i])    #считаем стоимость
                                      доставки по кругу
minim = cost
for i in range(1, n):
    cost = cost - s[i] + s[(i + n // 2) % n]  #подсчитываем
                                              стоимость доставки,
    if minim > cost:                          #передвигая точки
                                              старта
        minim = cost
        ans = i
print(ans+1)  #добавляем +1 в ответ, т.к.
              индексация с нуля
```

ОЧЕРЕДИ

```
#Количество пар, находящихся на расстоянии 5,
произведение которых четно.

f = open('27.txt')
n = int(f.readline())

line = []
for i in range(4):
    line.append(int(f.readline()))

ans = 0
even = 0
odd = 0
for i in range(4,n):
    x = int(f.readline())
    if x%2==0:
        ans+=odd+even
    else:
        ans+=even

    #сброс самого левого элемента на свалку
    if line[0]%2==0:
        even+=1
    else:
        odd+=1

    #сдвигаем очередь
    for j in range(3):
        line[j]=line[j+1]
    line[3]=x

print(ans)
```