Единый государственный экзамен
по ИНФОРМАТИКЕ и ИКТ

Инструкция по выполнению работы

Экзаменационная работа состоит из 27 заданий с кратким ответом, выполняемых с помощью компьютера.

На выполнение экзаменационной работы по информатике и ИКТ отводится 3 часа 55 минут (235 минут).

Экзаменационная работа выполняется с помощью специализированного программного обеспечения, предназначенного для проведения экзамена в компьютерной форме. При выполнении заданий Вам будут доступны на протяжении всего экзамена текстовый редактор, редактор электронных таблиц, системы программирования. Расположение указанного программного обеспечения на компьютере и каталог для создания электронных файлов при выполнении заданий Вам укажет организатор в аудитории.

На протяжении сдачи экзамена доступ к сети Интернет запрещён.

При выполнении заданий можно пользоваться черновиком. **Записи в черновике не учитываются при оценивании работы**.

Баллы, полученные Вами за выполненные задания, суммируются. Постарайтесь выполнить как можно больше заданий и набрать наибольшее количество баллов

Желаем успеха!

В экзаменационных заданиях используются следующие соглашения.

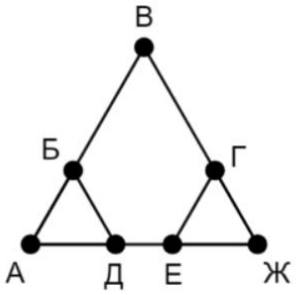
1. Обозначения для логических связок (операций):
   a) отрицание (инверсия, логическое НЕ) обозначается ¬ (например, ¬A);
   b) конъюнкция (логическое умножение, логическое И) обозначается /\ (например, A /\ B) либо & (например, A & B);
   c) дизъюнкция (логическое сложение, логическое ИЛИ) обозначается \/ (например, A \/ B) либо | (например, A | B);
   d) следование (импликация) обозначается → (например, A → B);
   e) тождество обозначается ≡ (например, A ≡ B). Выражение A ≡ B истинно тогда и только тогда, когда значения A и B совпадают (либо они оба истинны, либо они оба ложны);
   f) символ 1 используется для обозначения истины (истинного высказывания); символ 0 – для обозначения лжи (ложного высказывания).

2. Два логических выражения, содержащих переменные, называются равносильными (эквивалентными), если значения этих выражений совпадают при любых значениях переменных. Так, выражения A → B и (¬A) \/ B равносильны, а A \/ B и A /\ B неравносильны (значения выражений разные, например, при A = 1, B = 0).

3. Приоритеты логических операций: инверсия (отрицание), конъюнкция (логическое умножение), дизъюнкция (логическое сложение), импликация (следование), тождество. Таким образом, ¬A /\ B \/ C /\ D означает то же, что и ((¬A) /\ B) \/ (C /\ D).
Возможна запись A /\ B /\ C вместо (A /\ B) /\ C. То же относится и к дизъюнкции: возможна запись A \/ B \/ C вместо (A \/ B) \/ C.

4. Обозначения Мбайт и Кбайт используются в традиционном для информатики смысле – как обозначения единиц измерения, чьё соотношение с единицей «байт» выражается степенью двойки.

1 На рисунке схема дорог Н-ского района изображена в виде графа, в таблице содержатся сведения о протяжённости каждой из этих дорог (в километрах).



		Номер пункта						
		1	2	3	4	5	6	7
Номер пункта	1			3	4		10	
	2					9		8
	3	3			6			
	4	4		6				7
	5		9				11	8
	6	10				11		
	7		8		7	8		

Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какова протяжённость дороги из пункта Д в пункт Е. В ответе запишите целое число – так, как оно указано в таблице.

Решение (анализ графа и таблицы):

Заметим, что вершина Б и Г связаны с двумя вершинами, в которых есть только две дороги. У пунктов Д и Е такого свойства нет. Значит, мы можем найти пункты Д и Е перебрав все строки таблицы, в которых три числа. И выбрать из них те, дороги из которых приводят только в один пункт с двумя дорогами. Так как таблица симметричная, для этого мы можем проверять количество дорог в ячейках с числами.

Возьмем П1. Из него есть дороги в П3, П4 и П6. В столбце для П3 две дороги (значит из пункта П3 две дороги), для П4 – три дороги, для П6 – две дороги. Следовательно, это либо город Б, либо город Г.

Из П4 есть дороги в П1, П3 и П7. П1 – 3 дороги, П3 – 2 дороги, П7 – 3 дороги. Следовательно, это один из искомых городов Д или Е.

Из П5 есть дороги в П2, П6 и П7. Нет дороги в П4, следовательно это либо Б, либо Г.

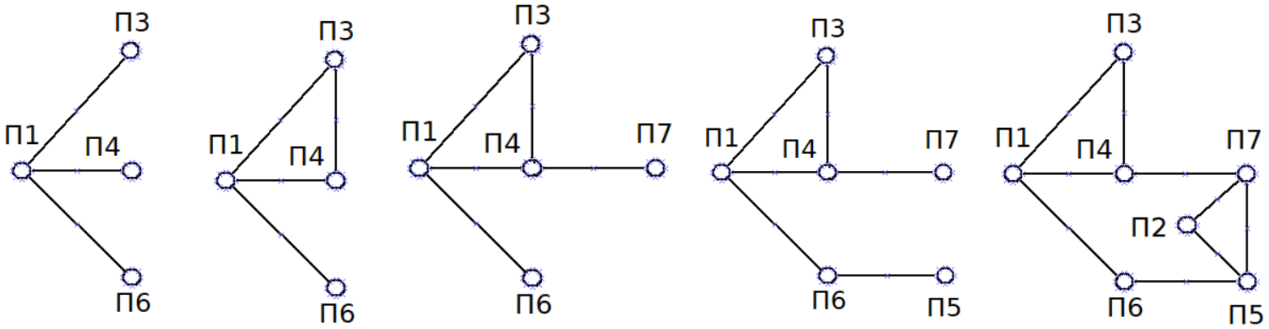
Из П7 (путем исключения второй искомый пункт) есть дороги в П2, П4, П5 (2, 3, 3 дорог - подтверждение).

Следовательно, необходимо найти длину дороги из П4 в П7. Это 7.

Ответ: 7

Решение (графическое):

Построим по таблице граф. Начнем с П1, есть дороги в П3, П4 и П7. И продолжим простраивать граф от полученных вершин.



По построенному графу видно два треугольника, которые есть на графе из задания, которые соединены через пункты П4 и П7. Следовательно, это и есть города Д и Е. Ответ 7.

Ответ: 7

Решение (программное):

С помощью переборного алгоритма найдем все возможные соответствия вершинам графа пунктов из таблицы. Граф представим как строку, где подстроки, разделенные пробелом, будут представлять собой пункт (первый символ) и пункты, в которые можно из него попасть, остальные символы.

Python

```
from itertools import permutations
s = 'абд бавд вбг гвеж дабе егдж жег'
s_g = {c[0]:set(c[1:]) for c in s.split()}
s1 = '1346 257 314 4137 5267 615 7245'
print('1234567')
for x in permutations(set(s) - {' '}):
    s2 = s1
    for a1, a2 in zip('1234567', x):
        s2 = s2.replace(a1, a2)
    s2_g = {c[0]: set(c[1:]) for c in s2.split()}
    if s2_g == s_g:
        print(*x, sep='')
```

PascalABC.net

```
// Автор: Максим Крючков
##
var m:='346 57 14 137 267 15 245'.split;
var d:='АБД БАВД ВБГ ГВЕЖ ДАБЕ ЕГДЖ ЖГЕ'
       .Split.ToDictionary(x->x[1],x->x[2:]);
foreach var xx in d.Keys.Permutations do
  if d.Keys.all(
      y-> d[y] = m[xx.IndexOf(y)]
                 .Select(z->xx[z.ToDigit-1])
                 .Sorted.JoinToString
  )
  then println(xx);
```

Получим вывод

```
1234567
гажебвд
бжадгве
```

Пункты Д и Е – пункты 4 и 7 (или 7 и 4). Следовательно, ответ 7.

Ответ: 7

2 Логическая функция F задаётся выражением (x ∧ ¬y) ∨ (y ≡ z) ∨ ¬w. На рисунке приведён фрагмент таблицы истинности функции F, содержащий неповторяющиеся наборы аргументов.

?	?	?	?	F
0			0	0
0	1	0	1	0
	1	0		0

Определите, какому столбцу таблицы соответствует каждая из переменных w, x, y, z.

В ответе запишите буквы w, x, y, z в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу и т.д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Решение (аналитическое):

Проанализируем выражение и таблицу. Видим три подвыражения перечисленные через знак дизъюнкции. Все выражение для трех различных наборов имеет ложные значения.

Составим таблицы истинности для всех подвыражений, когда они ложны.

x	**y**	**x ∧ ¬y**
0	0	0
0	1	0
1	1	0

y	**z**	**y ≡ z**
0	1	0
1	0	0

w	**¬w**
1	0

Из полученных таблиц истинности выведем все возможные наборы x, y, z, w для всего выражения.

x	y	z	w	F
0	0	1	1	0
0	1	0	1	0
1	1	0	1	0

Сопоставим полученный фрагмент с фрагментом из задания

?	?	?	?	F
0			0	0
0	1	0	1	0
	1	0		0

Заметим, что w может быть только вторым столбцом (все единицы). y – последний столбец (один ноль). z – третий столбец, так как противоположен по значению столбцу y (1 и 4 столбец – противоречие в первой строке – одинаковые значения). Следовательно, x – первый столбец.

Ответ: **xwzy**

Решение (программный метод нахождения фрагмента таблицы истинности)

Python	PascalABC.net

`from itertools import product`
`def f(x, y, z, w):`
`  return (x and not y) or (y == z) or not w`
 	
`print('x y z w')`
`for x, y, z, w in product([0,1], repeat=4):`
`  if f(x, y, z, w) == 0:`
`  print(x, y, z, w)`
	`// Автор: Максим Крючков`
	`## uses school;`
	`TrueTablePrint(`
	`  TrueTable((w,x,y,z)->`
	`  (x and not y) or (y=z) or not w`
	`  ), 0, 'wxyz');`

Получим следующий вывод

x y z w
0 0 1 1
0 1 0 1
1 1 0 1

Дальнейшие рассуждения аналогичны выше приведенным для аналитического решения.

Решение (программное, полное):

Данное решение построено на переборе всех возможных пропусков во фрагменте из задания и всех возможных перестановок имен столбцов.

Python	PascalABC.net

`def f(x, y, z, w):`
`  return (x and not y) or (y == z) or not w`
 	
`from itertools import *`
 	
`# перебор всех пробелов`
`for a1, a2, a3, a4 in product([0, 1], repeat=4):`
`  # все наборы`
`  t = [(0,a1,a2,0),(0,1,0,1),(a3,1,0,a4)]`
`  # проверка различности всех наборов`
`  if len(set(t)) != len(t):`
`  continue`
`  # перебор всех перестановок имен столбцов`
`  for p in permutations('xyzw'):`
`  # проверка на совпадение результата`
`  if [f(**dict(zip(p,r))) for r in t]==[0,0,0]:`
`  print(*p, sep='')`	

Ответ: **xwzy**

3 В файле приведён фрагмент базы данных «Текстиль» о поставках товаров в магазины районов города. База данных состоит из трёх таблиц. Таблица «Движение товаров» содержит записи о поставках товаров в магазины в течение первого полугодия 2023 г., а также информацию о проданных товарах. Поле Тип операции содержит значение Поступление или Продажа, а в соответствующее поле Количество упаковок, шт. занесена информация о том, сколько упаковок товара поступило в магазин или было продано в течение дня.

Заголовок таблицы имеет следующий вид.

ID операции	Дата	ID магазина	Артикул	Тип операции	Количество упаковок	Цена

Таблица «Товар» содержит информацию об основных характеристиках каждого товара. Заголовок таблицы имеет следующий вид.

Артикул	Отдел	Наименование	Единица измерения	Количество в упаковке	Производитель

Таблица «Магазин» содержит информацию о местонахождении магазинов. Заголовок таблицы имеет следующий вид.

ID магазина	Район	Адрес

На рисунке приведена схема указанной базы данных.

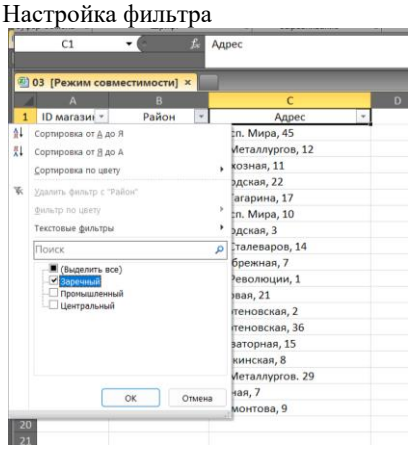
Используя информацию из приведённой базы данных, определите, на сколько увеличилось количество упаковок тульских пряников с начинкой, имеющихся в наличии в магазинах Заречного района, за период с 3 по 14 августа включительно.

В ответе запишите только число.

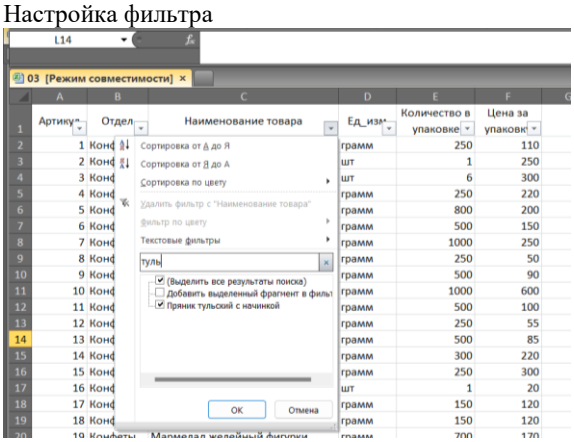
Решение (базовое):

Зайдем в таблицу «Магазин» и с помощью фильтра оставим только магазина Заречного района. Результат скопируем с помощью ножниц и вставим в таблице «Движение товаров». Также в таблице «Товары» найдем Артикул товара «Тульские пряники с начинкой».

Настройка фильтра



Настройка фильтра



Результат

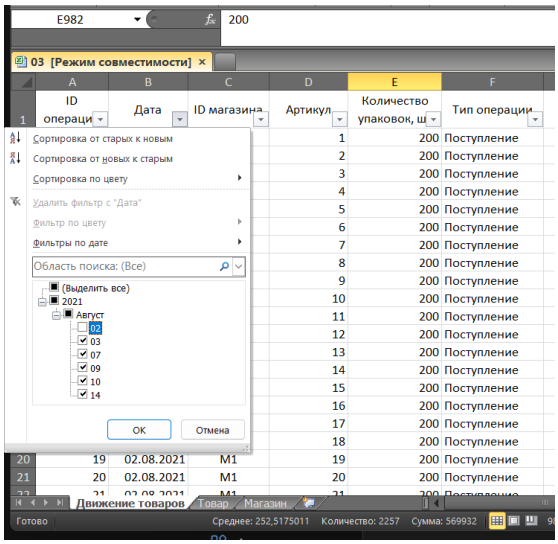
	A	B	C
1	ID магазин	Район	Адрес
4	M3	Заречный	Колхозная, 11
10	M9	Заречный	Прибрежная, 7
12	M11	Заречный	Луговая, 21
15	M14	Заречный	Элеваторная, 15
18	M17	Заречный	Лесная, 7

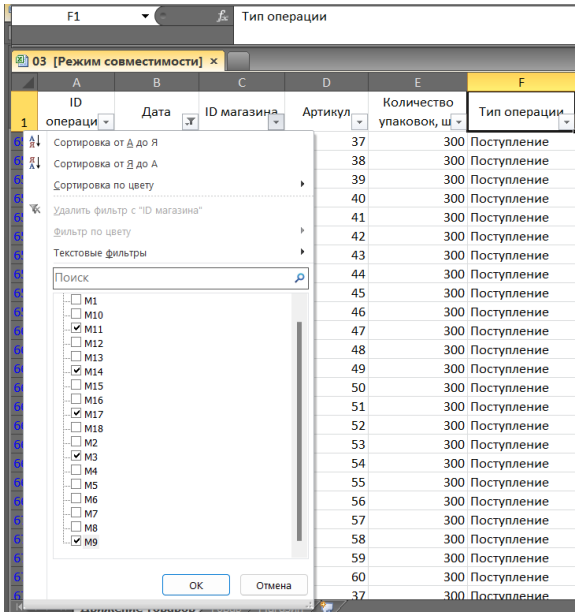
Результат

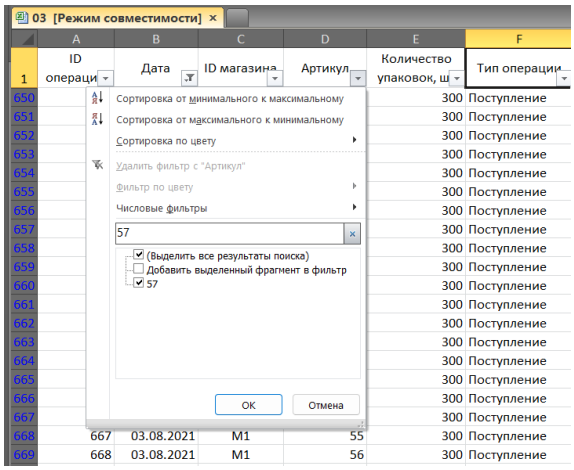
	A	B	C	D	E	F
1	Артику	Отдел	Наименование товара	Ед_изм	Количество в упаковке	Цена за упаковк
58	57	Печенье	Пряник тульский с начинкой	шт	1	40

Теперь в таблице «Движение товаров» наложим три фильтра «Дата» - 3 по 14 августа, «Артикул» = 57, «ID магазина» - M3, M9, M11, M14, M17.

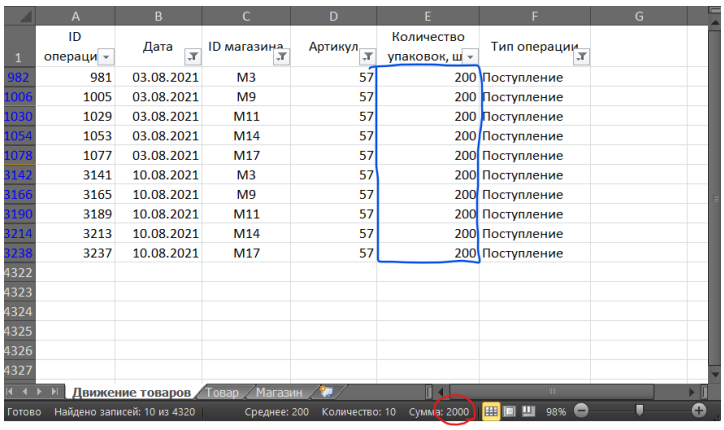
Настройки фильтров

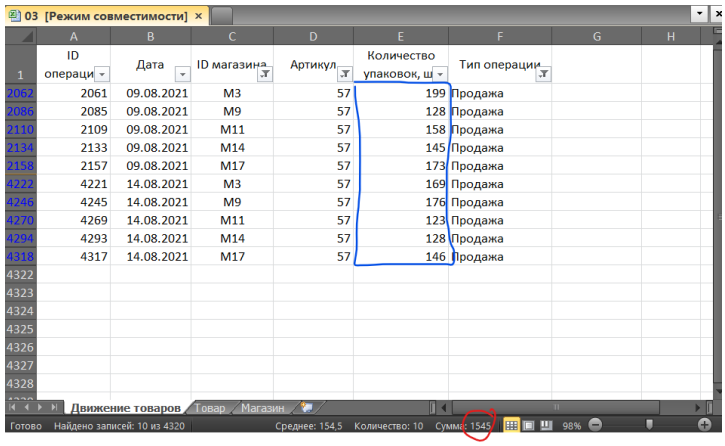






Чтобы узнать, насколько изменилось количество упаковок, узнаем (а) сколько упаковок поступило на склад и (б) сколько упаковок продано. После применения соответствующих фильтров, выделим столбец с количеством упаковок и в строке состояния увидим соответствующие количества.





Найдем разность 2000 и 1545.

Ответ: **455**

Решение (продвинутое, ВПР):

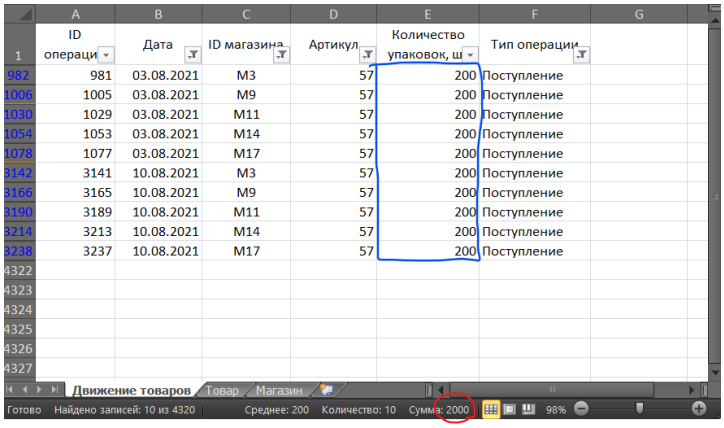
С помощью функции вертикального просмотра (ВПР) дополним таблицу «Движение товаров» двумя столбцами «Район» и «Товар».
Район для строки 2 определим, как =ВПР(C2;Магазин!A:C;2;0).
Товар, как =ВПР(D2;Товар!A:F;3;0)
Полученные формулы растянем на всю таблицу.

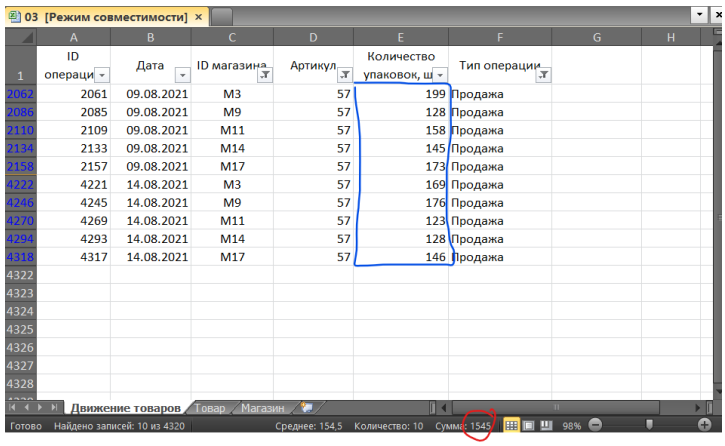
	A	B	C	D	E	F	G	H	I
1	ID операции	Дата	ID магазина	Артикул	Количество упаковок, шт	Тип операции	Район	Товар	
2	1	02.08.2021	M1	1	200	Поступление	Центральный	Батончик соевый	
3	2	02.08.2021	M1	2	200	Поступление	Центральный	Заяц шоколадный большой	
4	3	02.08.2021	M1	3	200	Поступление	Центральный	Заяц шоколадный малый	
5	4	02.08.2021	M1	4	200	Поступление	Центральный	Зефир в шоколаде	
6	5	02.08.2021	M1	5	200	Поступление	Центральный	Зефир ванильный	
7	6	02.08.2021	M1	6	200	Поступление	Центральный	Зефир воздушный	
8	7	02.08.2021	M1	7	200	Поступление	Центральный	Зефир лимонный	
9	8	02.08.2021	M1	8	200	Поступление	Центральный	Карамель "Барбарис"	
10	9	02.08.2021	M1	9	200	Поступление	Центральный	Карамель "Взлетная"	
11	10	02.08.2021	M1	10	200	Поступление	Центральный	Карамель "Раковая шейка"	
12	11	02.08.2021	M1	11	200	Поступление	Центральный	Карамель клубничная	
13	12	02.08.2021	M1	12	200	Поступление	Центральный	Карамель лимонная	

Теперь к полученной таблице применим фильтры по заданию: «Дата» - 3-14 августа, «Товар» - Пряник тульский с начинкой, «Район» - Заречный.

	A	B	C	D	E	F	G	H	I
1	ID операци	Дата	ID магазина	Артикул	Количество упаковок, ш	Тип операции	Район	Товар	
982	981	03.08.2021	M3	57	200	Поступление	Заречный	Пряник тульский с начинкой	
1006	1005	03.08.2021	M9	57	200	Поступление	Заречный	Пряник тульский с начинкой	
1030	1029	03.08.2021	M11	57	200	Поступление	Заречный	Пряник тульский с начинкой	
1054	1053	03.08.2021	M14	57	200	Поступление	Заречный	Пряник тульский с начинкой	
1078	1077	03.08.2021	M17	57	200	Поступление	Заречный	Пряник тульский с начинкой	
2062	2061	09.08.2021	M3	57	199	Продажа	Заречный	Пряник тульский с начинкой	
2086	2085	09.08.2021	M9	57	128	Продажа	Заречный	Пряник тульский с начинкой	
2110	2109	09.08.2021	M11	57	158	Продажа	Заречный	Пряник тульский с начинкой	
2134	2133	09.08.2021	M14	57	145	Продажа	Заречный	Пряник тульский с начинкой	
2158	2157	09.08.2021	M17	57	173	Продажа	Заречный	Пряник тульский с начинкой	
3142	3141	10.08.2021	M3	57	200	Поступление	Заречный	Пряник тульский с начинкой	
3166	3165	10.08.2021	M9	57	200	Поступление	Заречный	Пряник тульский с начинкой	
3190	3189	10.08.2021	M11	57	200	Поступление	Заречный	Пряник тульский с начинкой	
3214	3213	10.08.2021	M14	57	200	Поступление	Заречный	Пряник тульский с начинкой	
3238	3237	10.08.2021	M17	57	200	Поступление	Заречный	Пряник тульский с начинкой	
4222	4221	14.08.2021	M3	57	169	Продажа	Заречный	Пряник тульский с начинкой	
4246	4245	14.08.2021	M9	57	176	Продажа	Заречный	Пряник тульский с начинкой	
4270	4269	14.08.2021	M11	57	123	Продажа	Заречный	Пряник тульский с начинкой	

Чтобы узнать, насколько изменилось количество упаковок, узнаем (а) сколько упаковок поступило на склад и (б) сколько упаковок продано. После применения соответствующих фильтров, выделим столбец с количеством упаковок и в строке состояния увидим соответствующие количества.





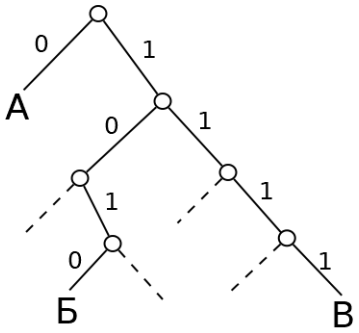
Найдем разность 2000 и 1545.

Ответ: **455**

4 По каналу связи передаются сообщения, содержащие только четыре буквы: А, Б, В, Г. Для передачи используется двоичный код, удовлетворяющий условию Фано. Кодовые слова для букв известны: А – 0, Б – 1111, В - 1010. Найдите код минимальной длины для буквы Г. Если таких кодов несколько, укажите код с минимальным числовым значением.

Примечание: условие Фано означает, что ни одно кодовое слово не является началом другого кодового слова.

Решение (построение двоичного дерева)



Заметим несколько свободных префиксов: 100, 1011, 110 и 1110. Выбираем префикс с минимальным числовым значением.

Ответ: **100**

Решение (перебор префиксов)
Рассмотрим все возможные префиксы длины до 4, пока не найдем подходящий.
Префиксы длиной 1: 0, 1. Оба префикса не удоблетворяют условию кодирования.
Префиксы длиной 2: 00, 01, 10, 11. 00 и 01 противоречит коду для А, 10 – для В, 11 – для Б.
Префиксы длиной 3: 000, 001, 010, 011, 100, 101, 110, 111. 000, 001, 010, 011 – противоречат А, 101 – В, 111 – Б. Префиксы 100 и 110 не противоречат уже существующим кодам. Наименьшее числовое значение из них имеет префикс 100.

Ответ: **100**

Решение (программный перебор префиксов)
Переберем префиксы программно (аналогично предыдущему решению)

Python

```
codes = ['0', '1010', '1111']
# длина префиксов от 1 до 4
for i in range(1, 5):
  # все префиксы, как двоичные числа
  for x in range(2**i):
    pfx = bin(x)[2:].zfill(i)
    # если префикс не является началом сущ.кода
    # и сущ.коды не являются началом префикса
    if all(c[:i] != pfx and pfx[:len(c)] != c
           for c in codes):
      print(pfx)
      break
```

PascalABC.net

```
// Автор: Максим Крючков
## uses school;
var alf:='0 1111 1010'.Split;
var kods:=(0..255).Select(x->Bin(x)).ToHashSet;
foreach var z in alf do
kods.RemoveWhere(
      s-> s.StartsWith(z) or z.StartsWith(s)
);
kods.OrderBy(s -> s.length)
    .ThenBy(s-> s).Take(1).Print;
```

Ответ: **100**

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 N кратно 3, тогда в конец дописывается три младших разряда полученной двоичной записи,
б) если число N не кратно 3, тогда в конец дописывается двоичная последовательность, являющаяся результатом умножения 3 на остаток от деления числа N на 3.
Полученная таким образом запись является двоичной записью искомого числа R.

Например, для исходного числа $5_{10} = 101_2$ результатом является число $101110_2 = 46_{10}$, а для исходного числа $9_{10} = 1001_2$ результатом является число $1001001_2 = 73_{10}$.

Укажите наибольшее число N, после обработки которого с помощью этого алгоритма получается число R, меньшее 100. В ответе запишите это число в десятичной системе счисления.

Решение (аналитическое):
Для аналитического рассуждения имеем два случая.
Число кратно 3.
Следовательно, число увеличивается в 8 раз (три двоичных раряда) и к нему дописывается значение до 7 (от 000 до 111).
100 // 8 = 12 (// - целочисленное деление)
12 делится на 3. Проверяем по алгоритму.
$12_{10} = 1100_2 \rightarrow 1100100_2 \rightarrow 100_{10}$ Следовательно, число 12 не удовлетворяет условию.
Проверим предыдущее, кратное 3 (9)
$9_{10} = 1001_2 = 1001001_2 = 73_{10}$
Следовательно, число 9 – последнее число, кратное 3, после обработки которого исполнитель получит значение меньшее 100.
Число не кратно 3
Для этого случая имеем два варианта – остаток равен 1 и остаток равен 2. В первом случае дописывается 11_2 (1*3), во втором – 110_2 (2*3).
- Остаток равен 2
  Дописывается три разряда, следовательно число увеличивается по формуле 8*x + 6. Значит, нужно взять число, меньшее 12, остаток от деления которого на 3 равен 2. Это число 11.
  11 * 8 + 6 = 94
  Следовательно, наибольшее число, остаток которого при делении на 3 равен 2, преобразуемое исполнителем в число меньшее 100, равно 11.
- Остаток равен 1
  Дописывается два разряда, следовательно число увеличивается по формуле 4*x + 3. Найдем максимальное число, которое может удовлетворять условию.
  100 // 4 = 25, 25 % 3 = 1 (остаток от деления)
  25*4 + 3 = 103 (не подходит)
  Берем предыдущее с таким же остатком.
  22*4 + 3 = 91.
  Следовательно, 22 – наибольшее число, остаток от деления на 3 у которого равен 1, и результат обработки испольнителем которого будет меньше 100.

Ответ: **22**

Решение (программное):

Python

```
for N in range(99, 0, -1):
    b = bin(N)[2:]
    if N % 3 == 0:
        b += b[-3:]
    else:
        b += bin(N % 3 * 3)[2:]
    if int(b, 2) < 100:
        print(N)
        break
```

PascalABC.net

```
### // Автор: Инна Свистун
uses school;
for var n:=4 to 100 do
begin
  var s:=bin(n);
  if (n mod 3=0) then s+=s[^3:]
  else s+=bin(3*(n mod 3));
if (dec(s,2)<100) then print(n);
end
```

Ответ: **22**

6 Исполнитель Черепаха передвигается по плоскости и оставляет след в виде линии. Черепаха может выполнять три команды:

По команде **Вперёд n** Черепаха перемещается вперёд на n единиц.

По команде **Направо m** Черепаха поворачивается на месте на m градусов по часовой стрелке, при этом соответственно меняется направление дальнейшего движения.

По команде **Налево m** Черепаха поворачивается на месте на m градусов против часовой стрелки, при этом соответственно меняется направление дальнейшего движения.

В начальный момент Черепаха находится в начале координат и направлена вверх (вдоль положительного направления оси ординат), хвост опущен.

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

Черепахе был дан для исполнения следующий алгоритм:

Направо 315

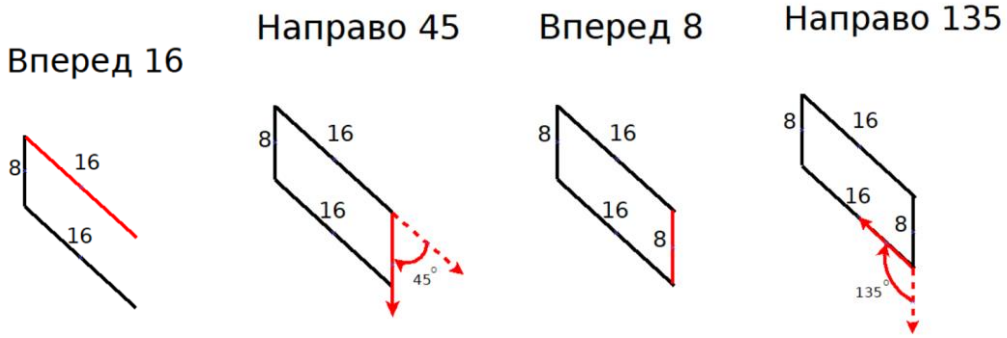
Повтори 7 [Вперёд 16 Направо 45 Вперёд 8 Направо 135]

Определите, сколько точек с целочисленными координатами будет находиться внутри фигуры, ограниченной заданным алгоритмом линиями, не включая точки на линиях.

Решение (аналитическое)

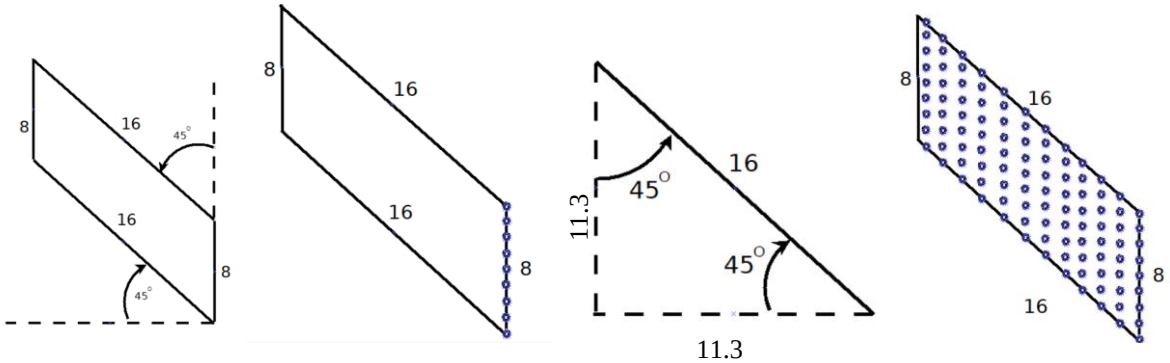
Схематически построим след черепахи, последовательно выполнив команды из алгоритма до конца первой итерации.

После первой итерации не находим замкнутую фигуру. Поэтому выполняем алгоритм для второй итерации.



Заметим, что исполнитель вернулся в точку, отеуда началась первая итерация. Следовательно, остальные 5 итераций будут просто повторять рисунок поверх уже нарисованного следа.

При выполнении алгоритма стоит помнить текущие углы отклонения. Так, например, после первой команды исполнитель будет отклонен от оси oX на 45 градусов, при следующем повороте отклонение будет на 90 градусов и т.д. Так же из курса геометрии мы знаем, что линия, проведенная под углом 45 градусов, начинающаяся в точке (x, y) с целочисленными координатами, проходит через все точки с кординатами (x+k, y+k), где k – целое число. Также мы можем подсчитать количество точек с целочисленными координатами на правой вертикальной стороне – точек всего 9. Теперь осталось найти количество вертикальных линий с шагом 1, которые пересекают фигуру. Для этого определим её длину через вычисления катетов равностороннего прямоугольного треугольника с гипотенузой длиной 16. $16^2 = 2x^2$, $128 = x^2$, $x = \sqrt{128} \approx 11.3$. Следовательно, количество точек внутри фигуры 7*11 = 77

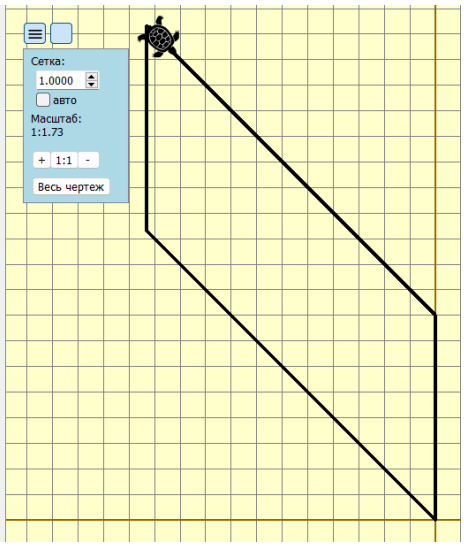


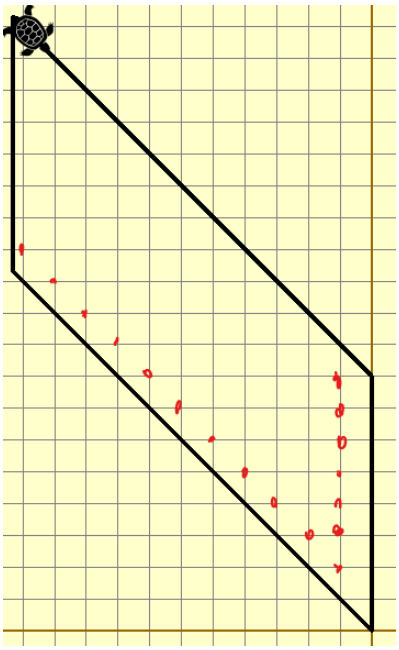
Ответ: 77

Решение (КуМИР)

Перенесем алгоритм из задания в среду КуМИР.

```
использовать Черепаха
алг
нач
. опустить хвост
. вправо(315)
. нц 7 раз
. . вперед(16)
. . вправо (45)
. . вперед(8)
. . вправо(135)
. кц
кон
```





Подсчитаем количество точек с целочисленными координатами. Заметим, что мы имеем 11 рядов по 7 точек.

Ответ: **77**

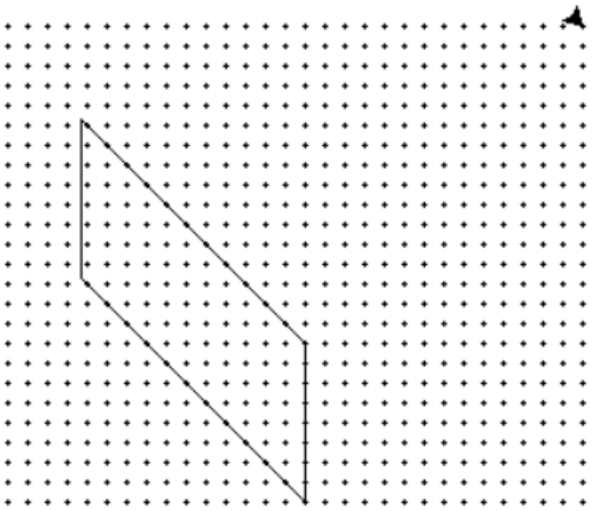
Решение (Python)

Воспользуемся библиотекой turtle, предварительно направив голову черепахи вдоль оси ординат. Добавив две команды для быстрой отрисовки – tracer(0) в начало алгоритма и update() в конец. Для большего масштаба увеличим каждое перемещение в 10 раз. После чего нанесем точки с целочисленными координатами. Так же не забудем унести команду update() в конец алгоритма.

```
from turtle import *
tracer(0)
left(90)

down()
right(315)
for _ in range(7):
    forward(16*10)
    right(45)
    forward(8*10)
    right(135)

for x in range(-15, 15):
    for y in range(0, 25):
        up()
        goto(x*10, y*10)
        down()
        dot(3)
update()
```



Подсчитаем количество точек внутри полученного контура.

Ответ: **77**

7 Голосовое сообщение, записанное в стерео формате, передается со скоростью 64000 бит/с. Файл был записан с такими параметрами: глубина кодирования – 24 бит на отсчет, частота дискретизации – 16000 отсчетов в секунду, время записи – 90 с. Сколько секунд будет передаваться голосовое сообщени?

Решение:
Сначала определим размер файла. Для этого перемножим значения: количество каналов, частота дискретизаци, время записи и глубину кодирования.

$$V = n \cdot v \cdot t \cdot I$$
$$V = 2 \cdot 16000 \cdot 90 \cdot 24 = 69120000 \text{ бит}$$

Теперь разделим размер файла на скорость передачи по каналу связи.

$$t = \frac{2 \cdot 16000 \cdot 90 \cdot 24}{64000} = 1080 \text{ секунд}$$

Ответ: **1080**

8 Все четырехбуквенные слова, в составе которых могут быть только русские буквы А, В, Л, О, Р записаны в алфавитном порядке и пронумерованы начиная с 1.
Ниже приведено начало списка.
1. АААА
2. АААВ
3. АААЛ
4. АААО
5. АААР
6. ААВА

Под каким номером идет первое слово, начинающееся на Л?

Решение (комбинаторное)
После анализа списка понимаем, что Л – третья буква, если располагать приведенные буквы в алфавитном порядке. Значит, для букв А и В будет записано по 5^3 комбинаций (все допустимые варианты для 2, 3 и 4 букв). Следовательно, Первое слово, начинающееся на Л, будет стоять на позиции 250 (количество комбинаций с первыми А и В) + 1.

Ответ: **251**

Решение (системы счисления)
Для удобства можем представить наши комбинации, как числа в пятеричной системе счисления, где буквы А, В, Л, О, Р можно сопоставить с цифрами 0, 1, 2, 3, 4 соответственно.
Значит, необходимо найти наименьшее число длиной 4, которое начинающееся на Л. Это число $2000_5 = 2*5^3 = 250$.
Так как список начинается с нулевой комбинации (АААА = 0000 = 0), то позиция числа в списке на единицу больше, чем само значение записанного числа.

Ответ: **251**

Решение (программное)

Python

```
from itertools import product
# перебираем все размещения с повторениями
# длиной 4 и нумеруем их с 1
for i, w in enumerate(product(sorted('АВРОЛ'),
                              repeat=4),
                      start=1):
    # выводим номер первого подходящего слова
    if w[0] == 'Л':
        print(i)
        break
```

```
from itertools import product
words = [''.join(x)
         for x in product('АВЛОР', repeat=4)]
print(words.index('ЛААА') + 1)
```

PascalABC.net

```
// Автор: Максим Крючков
##
'АВЛОР'.Cartesian(4).Order.Numerate
       .Where(\(n,s) -> s[1]='Л').First
       .Print;
```

Ответ: **251**

Задание выполняется с использованием прилагаемых файлов.

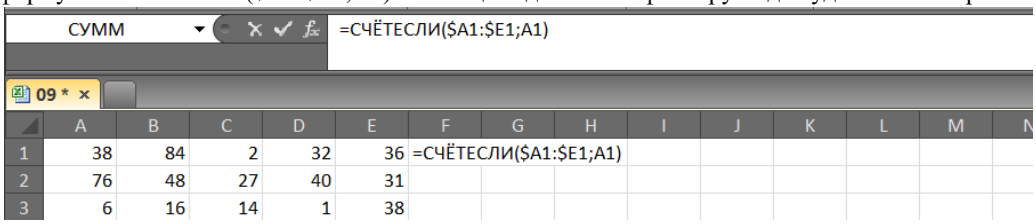
9 Откройте файл электронной таблицы, содержащей в каждой строке пять натуральных чисел.
Определите количество строк таблицы, содержащих числа, для которых выполнены оба условия:

- каждое число в строке встречается по одному разу,
- утроенная сумма максимального и минимального значений не превышает удвоенной суммы оставшихся чисел.

В ответе запишите только число.

Решение (через работу с максимумом и минимумом)

Через счётесли определим сколько раз в строке встречается каждое из чисел. Например, для числа в ячейке A1 будет следующая формула =СЧЁТЕСЛИ($A1:$E1;A1). Столбцы в диапазоне фиксируем для удобства копирования формулы для остальных 4 ячеек.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	38	84	2	32	36	=СЧЁТЕСЛИ($A1:$E1;A1)								
2	76	48	27	40	31									
3	6	16	14	1	38									

Можно заметить, что нам нужны строки либо с 5 единицами, либо с суммой 5 (что одно и то же, конечно же). Поэтому внесем формулу =сумм(F1:J1)=5

	A	B	C	D	E	F	G	H	I	J	K
1	38	84	2	32	36	1	1	1	1	1	ИСТИНА
2	76	48	27	40	31	1	1	1	1	1	ИСТИНА
3	6	16	14	1	38	1	1	1	1	1	ИСТИНА
4	92	28	19	33	38	1	1	1	1	1	ИСТИНА
5	7	86	35	23	11	1	1	1	1	1	ИСТИНА
6	74	99	74	27	38	2	1	2	1	1	ЛОЖЬ
7	42	5	77	18	49	1	1	1	1	1	ИСТИНА
8	32	84	28	18	2	1	1	1	1	1	ИСТИНА
9	27	7	38	28	3	1	1	1	1	1	ИСТИНА
10	51	29	71	15	25	1	1	1	1	1	ИСТИНА
11	47	56	49	40	31	1	1	1	1	1	ИСТИНА
12	66	56	64	13	11	1	1	1	1	1	ИСТИНА
13	95	82	15	42	30	1	1	1	1	1	ИСТИНА

Первое условие мы описали. Перейдем ко второму.

Для определения минимального и максимального числа в строке воспользуемся функциями МИН и МАКС. Сумму остальных трех чисел определим, как сумму всех чисел строки за исключением минимального и максимального значений.

Формула для поиска суммы минимального и максимального значений (ячейка L1): =МИН(A1:E1)+МАКС(A1:E1)

Формула для поиска суммы трех оставшихся (ячейка M1): =СУММ(A1:E1)-L1

Формула для второго условия задания: =3*L1 <= 2*M1

Для удобства поиска совпадений по двум условиям умножим значения в столбце K на значения в столбце N (так логические значения преобразуются в числовые и мы получим 1 в тех случаях, когда оба значения истинны). После чего найдем сумму значений в новом столбце.

Формула в столбце O: =K1*N1

Формула для получения ответа: =СУММ(O:O)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	38	84	2	32	36	1	1	1	1	1	ИСТИНА	86	106	ЛОЖЬ	0	853
2	76	48	27	40	31	1	1	1	1	1	ИСТИНА	103	119	ЛОЖЬ	0	
3	6	16	14	1	38	1	1	1	1	1	ИСТИНА	39	36	ЛОЖЬ	0	
4	92	28	19	33	38	1	1	1	1	1	ИСТИНА	111	99	ЛОЖЬ	0	
5	7	86	35	23	11	1	1	1	1	1	ИСТИНА	93	69	ЛОЖЬ	0	
6	74	99	74	27	38	2	1	2	1	1	ЛОЖЬ	126	186	ЛОЖЬ	0	
7	42	5	77	18	49	1	1	1	1	1	ИСТИНА	82	109	ЛОЖЬ	0	
8	32	84	28	18	2	1	1	1	1	1	ИСТИНА	86	78	ЛОЖЬ	0	
9	27	7	38	28	3	1	1	1	1	1	ИСТИНА	41	62	ИСТИНА	1	
10	51	29	71	15	25	1	1	1	1	1	ИСТИНА	86	105	ЛОЖЬ	0	
11	47	56	49	40	31	1	1	1	1	1	ИСТИНА	87	136	ИСТИНА	1	
12	66	56	64	13	11	1	1	1	1	1	ИСТИНА	77	133	ИСТИНА	1	
13	95	82	15	42	30	1	1	1	1	1	ИСТИНА	110	154	ЛОЖЬ	0	
14	3	22	78	16	5	1	1	1	1	1	ИСТИНА	81	43	ЛОЖЬ	0	
15	95	31	92	31	27	1	2	1	2	1	ЛОЖЬ	122	154	ЛОЖЬ	0	
16	93	9	47	21	48	1	1	1	1	1	ИСТИНА	102	116	ЛОЖЬ	0	
17	45	48	67	25	24	1	1	1	1	1	ИСТИНА	91	118	ЛОЖЬ	0	

Ответ: **853**

Решение (через НАИБОЛЬШЕЕ/НАИМЕНЬШЕЕ)

Суть этого решения в первоначальном упорядочивании элементов строки.

Так, вписав формулы =НАИБОЛЬШИЙ(A1:E1, 1), =НАИБОЛЬШИЙ(A1:E1, 2), =НАИБОЛЬШИЙ(A1:E1, 3), =НАИБОЛЬШИЙ(A1:E1, 4), =НАИБОЛЬШИЙ(A1:E1, 5) в ячейки F1:J1 соответственно, получим пять чисел, идущих в порядке убывания. Где значение в F1 будет максимальным значением, значение в J1 – минимальным.

09 * ×

	A	B	C	D	E	F	G	H	I	J
1	38	84	2	32	36	84	38	36	32	2
2	76	48	27	40	31	76	48	40	31	27
3	6	16	14	1	38	38	16	14	6	1
4	92	28	19	33	38	92	38	33	28	19
5	7	86	35	23	11	86	35	23	11	7
6	74	99	74	27	38	99	74	74	38	27
7	42	5	77	18	49	77	49	42	18	5
8	32	84	28	18	2	84	32	28	18	2
9	27	7	38	28	3	38	28	27	7	3
10	51	29	71	15	25	71	51	29	25	15
11	47	56	49	40	31	56	49	47	40	31
12	66	56	64	13	11	66	64	56	13	11

Теперь для проверки одинаковости (уникальности) чисел достаточно сравнить только пары рядом стоящих чисел.

Для первой строки формула: =И(F1>G1;G1>H1;H1>I1;I1>J1)

Для проверки второго условия: сумма максимума и минимума – F1+J1, сумма трех оставшихся – СУММ(G1:I1).

Значит, формула будет выглядеть, как = 3*(F1+J1)<=2*СУММ(G1:I1)

Для проверки совпадения двух условий так же воспользуемся формулой с умножением =K1*L1. Для нахождения ответа - =сумм(M:M)

09 * ×

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	38	84	2	32	36	84	38	36	32	2	ИСТИНА	ЛОЖЬ	0	853
2	76	48	27	40	31	76	48	40	31	27	ИСТИНА	ЛОЖЬ	0	
3	6	16	14	1	38	38	16	14	6	1	ИСТИНА	ЛОЖЬ	0	
4	92	28	19	33	38	92	38	33	28	19	ИСТИНА	ЛОЖЬ	0	
5	7	86	35	23	11	86	35	23	11	7	ИСТИНА	ЛОЖЬ	0	
6	74	99	74	27	38	99	74	74	38	27	ЛОЖЬ	ЛОЖЬ	0	
7	42	5	77	18	49	77	49	42	18	5	ИСТИНА	ЛОЖЬ	0	
8	32	84	28	18	2	84	32	28	18	2	ИСТИНА	ЛОЖЬ	0	
9	27	7	38	28	3	38	28	27	7	3	ИСТИНА	ИСТИНА	1	
10	51	29	71	15	25	71	51	29	25	15	ИСТИНА	ЛОЖЬ	0	
11	47	56	49	40	31	56	49	47	40	31	ИСТИНА	ИСТИНА	1	
12	66	56	64	13	11	66	64	56	13	11	ИСТИНА	ИСТИНА	1	
13	95	82	15	42	30	95	82	42	30	15	ИСТИНА	ЛОЖЬ	0	
14	3	22	78	16	5	78	22	16	5	3	ИСТИНА	ЛОЖЬ	0	
15	95	31	92	31	27	95	92	31	31	27	ЛОЖЬ	ЛОЖЬ	0	
16	93	9	47	21	48	93	48	47	21	9	ИСТИНА	ЛОЖЬ	0	
17	45	48	67	25	24	67	48	45	25	24	ИСТИНА	ЛОЖЬ	0	
18	88	10	90	47	36	90	88	47	36	10	ИСТИНА	ИСТИНА	1	

Ответ: **853**

Решение (программное)

Сохраним массив данных в текстовом файле (я сохранял из excel в формате csv с разделителями по точке с запятой, можно просто скопировать и вставить в блокноте все числа).

Python

```
with open('09.csv') as f:
  # если просто копировать и вставить
  # можно не делать замену (replace)
  # сортировка сразу для удобства
  nums = [sorted(map(int, s.replace(';', ' ')
                          .split()))]
          for s in f]
cnt = 0
for t in nums:
  if len(set(t)) == len(t) \
     and 3*(t[0]+t[-1]) <= 2*sum(t[1:4]):
    cnt += 1
print(cnt)
```

PascalABC.net

Задание выполняется с использованием прилагаемых файлов.

10 Текст Повести Александра Куприна «Поединок» представлен в виде файлов различных форматов. Откройте один из файлов и определите, сколько раз в тексте встречаются комбинация символов «Час» или «час», не являющиеся отдельными словами.

В ответе запишите только число.

Решение (через меню замены)
Вызываем мены замены (MS Word и LibreOffice – Ctrl+H). Устанавливаем флаги «слово целиком» и «с учетом регистра». Данные флаги нужны для того, чтобы найти слова, которые не нужно учитывать. И, так как нам даны конкретные комбинации символов, нужно учитывать регистр (решение с заменой без учета регистра комбинаций «час» будет нестрогим решением этой задачи). Сначала заменяем комбинацию «час» на пробел, затем комбинацию «Час» на пробел (можно на любую другую последовательность, не содержащую данные комбинации). Ради интереса можем заметить что было произведено 6 замен (5 для первого случая и 1 для второго).

Теперь уберем флаг «слово целиком» и проведем такую же манипуляцию по замене входящих в другие слова указанных в задании комбинаций. Увидим 262 замены для комбинации «час» и 9 замен для комбинации «Час».

Ответ: **271**

11 При регистрации в компьютерной системе каждому объекту присваивается идентификатор, состоящий из 113 символов и содержащий только десятичные цифры и символы из 2025-символьного специального алфавита. В базе данных для хранения каждого идентификатора отведено одинаковое и минимально возможное целое число байт. При этом используют посимвольное кодирование идентификаторов, все символы кодируют одинаковым и минимально возможным количеством бит.

Определите объём памяти (в Кбайт), необходимый для хранения 32 768 идентификаторов.

В ответе запишите только целое число – количество Кбайт.

Решение
Определим длину алфавита, используемого для записи идентификатора.

$$|A| = 10 + 2025 = 2035$$

Определим количество бит, необходимое для хранения одного из 2035 символов ($\lceil x \rceil$ - операция округления значения x вверх до целого)

$$I = \lceil \log_2 2035 \rceil = 11$$

Или воспользуемся обратной формулой, значение I в которой найдем в целых числах.

$$2^I \geq 2035$$
$$I_{min} = 11$$

Теперь определим целое количество байт, необходимое для хранения 113 символов идентификатора.

$$S = \left\lceil \frac{113 \cdot 11}{8} \right\rceil = \lceil 155.375 \rceil = 156$$

Наконец найдем количество Кбайт для хранения 32768 идентификаторов.

$$\frac{156 \cdot 32768}{1024} = 4992$$

Ответ: **4992**

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) или команда2 (если условие ложно).
Исполнитель Редактор получает на вход строку начинающуюся на 3 и содержащую далее n пятерок (5), где n > 3.
На выполнение Редактору дана следующая программа:

```
НАЧАЛО
ПОКА нашлось (25) ИЛИ нашлось (355) ИЛИ нашлось (555)
    ЕСЛИ нашлось (25)
      ТО заменить (25, 3)
    КОНЕЦ ЕСЛИ
    ЕСЛИ нашлось (355)
      ТО заменить (355, 52)
    КОНЕЦ ЕСЛИ
    ЕСЛИ нашлось (555)
      ТО заменить (555, 23)
    КОНЕЦ ЕСЛИ
КОНЕЦ ПОКА
КОНЕЦ
```

Найдите минимальное значение n, при котором сумма цифр конечной строки будет равна 27.

Решение (аналитическое)

Последовательно рассмотрим преобразования строк. В скобках будет указывать номер срабатываемого условия

9 пятерок	**10 пятерок**	**11 пятерок**	**12 пятерок**	**13 пятерок**	**14 пятерок**
3555555555 1 итерация 525555555 (2) 52235555 (3) 2 итерация 5225255 (2) 3 итерация 523255 (1) 4 итерация 52335 (1)	Разница отсюда 52252555 (2) 5225223 (3) 3 итерация 523223 (1)	Разница отсюда 522525555 (2) 52252235 (3) 3 итерация 5232235 (1)	Разница отсюда 5225255555 (2) 522522355 (3) 3 итерация 52322355 (1) 5232252(2) 4 итерация 523232 (1)	Разница отсюда 52252555555 (2) 5225223555 (3) 3 итерация 523223555 (1) 52322525(2) 4 итерация 5232325 (1) 5 итерация 523233 (1)	Разница отсюда 522525555555 (2) 52252235555 (3) 3 итерация 5232235555 (1) 523225255(2) 4 итерация 52323255 (1) 5 итерация 5232335 (1)

Заметим, что при исследовании строки с 14 пятерками мы получили строку аналогичной конфигурации, как и после обработки строки из 9 пятерок. Следовательно, с шагом 5 мы будем получать строки конфигурации аналогичной значению n на 5 меньше, в которой после 5 будет идти на одну пару 23 больше.

Суммы для n:

9: 18	10: 17	11: 22	12: 17	13: 23
14: 23	15: 22	16: 27		

Ответ: **16**

Примечание: в этой задаче можно было начать исследование и с n=3, но на малых значениях изменение строки может не проходить по общему правилу, поэтому лучше исследовать строки бОльшей длины для выявления закономерности.

Решение (программное):

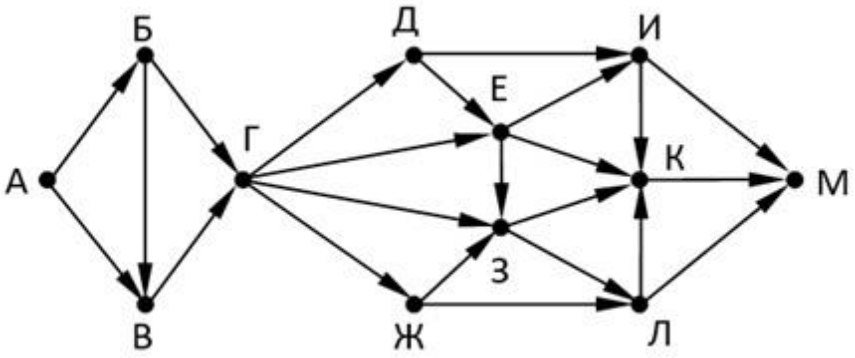
Python

```
for i in range(3, 100):
  s = '3' + '5'*i
  while '25' in s or '355' in s or '555' in s:
    s = s.replace('25', '3', 1)
    s = s.replace('355', '52', 1)
    s = s.replace('555', '23', 1)
  if sum(map(int, s)) == 27:
    print(i)
    break
```

PascalABC.net

```
// Автор: Максим Крючков
##
for var n:=4 to 50 do begin
var s:='3'+'5'*n;
while ('25' in s)or('355' in s)or('555' in s) do
  begin
    if '25' in s then s:=s.Replace('25', '3', 1);
    if '355' in s then s:=s.Replace('355','52', 1);
    if '555' in s then s:=s.Replace('555','23', 1);
  end;
if s.Sum(s-> s.ToDigit).Divs(27) then println(n);
end;
```

13 На рисунке представлена схема дорог, связывающих города А, Б, В, Г, Д, Е, Ж, З, И, К, Л, М. По каждой дороге можно двигаться только в одном направлении, указанном стрелкой. Определите количество различных путей, которые начинаются в городе А и заканчиваются в городе М.



Решение (ручное, рекурсивное, от «А»)

Решение строится на основе наблюдения, что количество путей в любую вершину равно сумме количеств путей в предшествующие вершины. Например, в Г есть 3 маршрута – АБГ, АВГ и АБВГ. Это один маршрут через Б и два маршрута через В. В Б есть один маршрут – АБ, в В есть два маршрута – АВ и АБВ. Заметим, что если дописать к этим трем маршрутам Г, то получим все маршруты из Г. Так же данный алгоритм можно описать, как нахождение количества маршрутов из начальной вершины (так как после выполнения алгоритма будут найдены соответствующие значения для всех вершин графа).

Поэтому рассчитываем количество маршрутов по рекуррентным формулам.

F(М) = F(И) + F(К) + F(Л)
F(К) = F(И) + F(Е) + F(З) + F(Л)
F(И) = F(Д) + F(Е)
F(Л) = F(З) + F(Ж)
F(З) = F(Е) + F(Г) + F(Ж)
F(Е) = F(Д) + F(Г)
F(Д) = F(Г)
F(Ж) = F(Г)
F(Г) = F(Б) + F(В)
F(В) = F(А) + F(Б)
F(Б) = F(А)
F(А) = 1

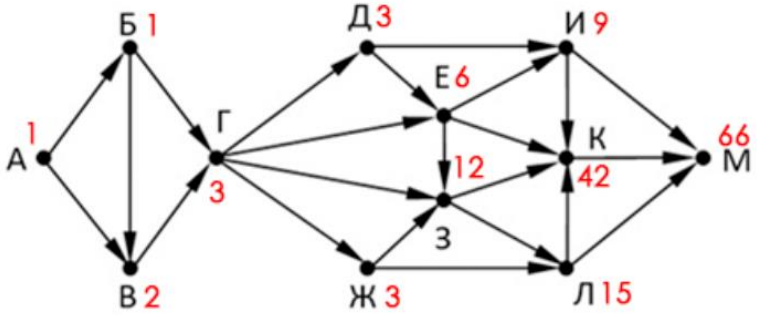
Теперь запускаем «обратных ход рекурсии», подсчитывая все значения, данные для вычисления которых уже получены.

F(Б) = F(А) = 1
F(В) = F(А) + F(Б) = 1 + 1 = 2
F(Г) = F(Б) + F(В) = 1 + 2 = 3
F(Ж) = F(Г) = 3
F(Д) = F(Г) = 3
F(Е) = F(Д) + F(Г) = 3 + 3 = 6
F(З) = F(Е) + F(Г) + F(Ж) = 6 + 3 + 3 = 12
F(Л) = F(З) + F(Ж) = 12 + 3 = 15
F(И) = F(Д) + F(Е) = 3 + 6 = 9
F(К) = F(И) + F(Е) + F(З) + F(Л) = 9 + 6 + 12 + 15 = 42
F(М) = F(И) + F(К) + F(Л) = 9 + 42 + 15 = 66

Ответ: **66**

Решение (ручное, графическое, от «А»)

Данное решение дублирует рассуждение из предыдущего приведенного решения, где сразу восстанавливается обратный ход рекурсии.



Решение (программное, циклическое, от «А»)

Принцип решения:

1) Описываем все входящие дуги
2) Определяем значение в А, как 1
3) Если значения для всех начальных пунктов во входящих дугах определены, то находим значение в текущей вершине
4) Выполняем алгоритм до тех пор, пока значения для всех вершин не будет подсчитано.

Python	PascalABC.net

`g_str = 'ба вба гвб дг едг жг згеж иде киезл лзж микл'`
`g = {c[0]: c[1:] for c in g_str.split()}`
`r = {'а': 1}`
`while not all(x in r for x in g):`
`  for v, d in g.items():`
`  if v not in r and all(x in r for x in d):`
`  r[v] = sum(r[x] for x in d)`
`print(r['м'])`	

Решение (программное, рекурсивное, от «А»)

Python	PascalABC.net

`g_str = 'ба вба гвб дг едг жг згеж иде киезл лзж микл'`
`g = {c[0]: c[1:] for c in g_str.split()}`
`def f(v):`
`  if v == 'а':`
`  return 1`
`  return sum(f(x) for x in g[v])`
`print(f('м'))`
	`## // Автор: Алексей Богданов`
	`var d:='А БА ВАБ ГБВ ДГ ЕГД ЖГ ЗГЖЕ ИДЕ КИЕЗЛ ЛЖЗ МИКЛ'.Split.ToDictionary(w->w.first,w->w?[2:]);`
	
	`var f:char->integer;`
	`f:=a-> a='A' ? 1: d[a].Sum(c->f(c));`
	`print(f('M'));`

Решение (ручное, аналитическое, движение в «М»)

Рассуждение очень похоже на рассуждение при движении от А. Разница заключается в том, что мы постепенно находим для каждой вершины количество маршрутов из этих вершин в конечную вершину. Например, из З можно попасть в М тремя способами – ЗЛМ, ЗЛКМ и ЗКМ. Причем двумя способами из Л (ЛКМ и ЛМ – все маршруты из Л в М), и одним из К (собственно, из К в М только один маршрут).

Поэтому имеем следующие рекуррентные отношения

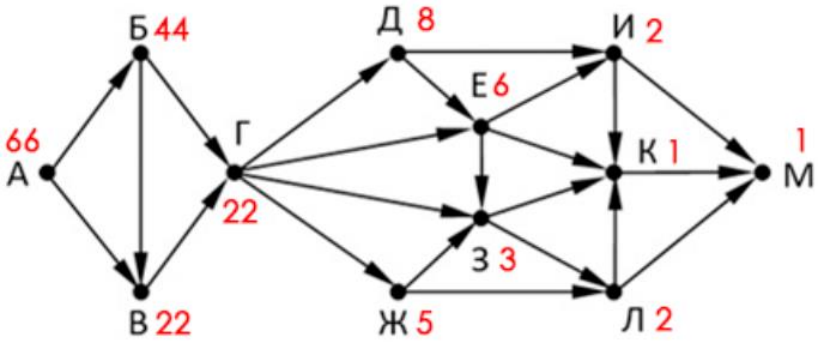
F(А) = F(Б) + F(В); F(Б) = F(В) + F(Г); F(В) = F(Г); F(Г) = F(Д) + F(Е) + F(Ж) + F(З); F(Д) = F(Е) + F(И); F(Е) = F(И) + F(К) + F(З)
F(Ж) = F(З) + F(Л); F(З) = F(К) + F(Л); F(И) = F(К) + F(М); F(К) = F(М); F(Л) = F(К) + F(М); F(М) = 1

Теперь рекурсивно восстановим все значения.

F(К) = F(М) = 1
F(Л) = F(К) + F(М) = 1 + 1 = 2
F(И) = F(К) + F(М) = 1 + 1 = 2
F(З) = F(К) + F(Л) = 1 + 2 = 3
F(Ж) = F(З) + F(Л) = 3 + 2 = 5
F(Е) = F(И) + F(К) + F(З) = 2 + 1 + 3 = 6
F(Д) = F(Е) + F(И) = 6 + 2 = 8
F(Г) = F(Д) + F(Е) + F(Ж) + F(З) = 8 + 6 + 5 + 3 = 22
F(В) = F(Г) = 22
F(Б) = F(В) + F(Г) = 22 + 22 = 44
F(А) = F(Б) + F(В) = 44 + 22 = 66

Ответ: **66**

Решение (ручное, графическое, движение в «М»)



Решение (программное, циклическое, в «М»)

Принцип решения:

1) Описываем все входящие дуги
2) Определяем значение в М, как 1
3) Если значения для всех начальных пунктов во входящих дугах определены, то находим значение в текущей вершине
4) Выполняем алгоритм до тех пор, пока значения для всех вершин не будет подсчитано.

Python	PascalABC.net

```
g_str = 'абв бгв вг гдежз деи еикз жзл зкл икм км лкм'
g = {c[0]: c[1:] for c in g_str.split()}
r = {'м': 1}
while not all(x in r for x in g):
    for v, d in g.items():
        if v not in r and all(x in r
                              for x in d):
            r[v] = sum(r[x] for x in d)
print(r['a'])
```

Решение (программное, рекурсивное, в «М»)

Python	PascalABC.net

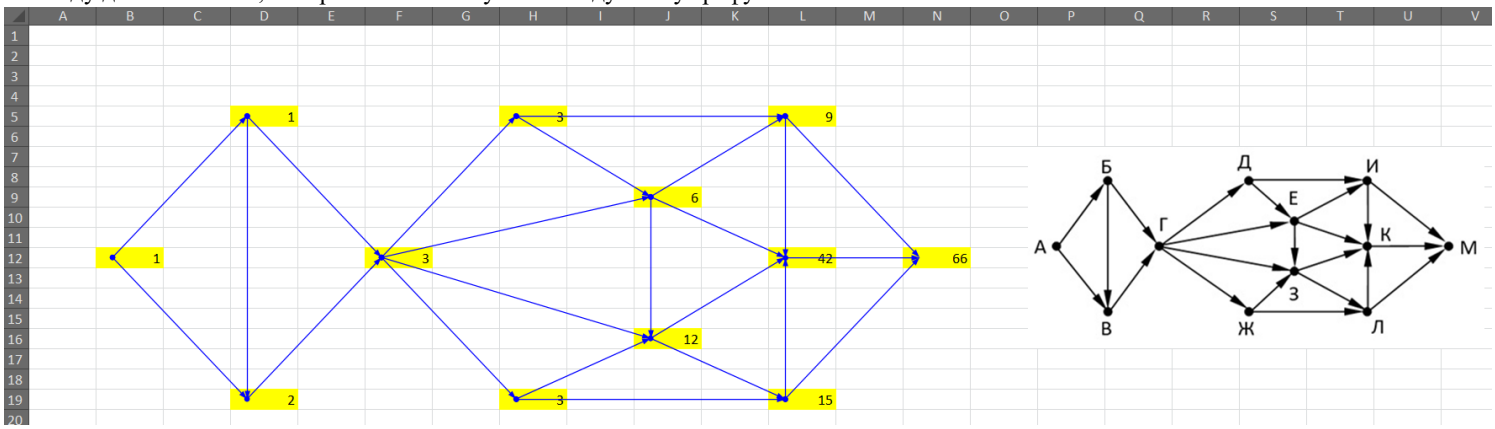
```
g_str = 'абв бгв вг гдежз деи еикз жзл зкл икм км лкм'
g = {c[0]: c[1:] for c in g_str.split()}
def f(v):
    if v == 'м':
        return 1
    return sum(f(x) for x in g[v])
print(f('a'))
```

Решение (графически-автоматизированное, электронные таблицы, прямой проход)

Предварительно подготовим таблицу, раскрасив ячейки в которых будем подсчитывать значения.

Теперь запишем в соответствующие ячейки формулы для нахождения количества путей (аналогично первому ручному решению). Например, для вершины М получим такую формулу N12**=L5+L12+L19**

Для удобства проверки можно воспользоваться функцией «Формулы»-«Зависимость формул»-«Зависимые ячейки», применив эту команду для всех ячеек, которые соответствуют исследуемому графу.



Ответ: **66**

14 Операнды арифметического выражения записаны в системе счисления с основанием 15.

$$97968x13_{15} + 7x213_{15}$$

В записи чисел переменной x обозначена неизвестная цифра из алфавита 15-ричной системы счисления. Определите наименьшее значение x, при котором значение данного арифметического выражения кратно 14. Для найденного значения x вычислите частное от деления значения арифметического выражения на 14 и укажите его в ответе в десятичной системе счисления. Основание системы счисления в ответе указывать не нужно.

Решение (аналитическое)

Так как нам необходимо найти число, кратное 14, исследуем разряды числа на соответствие этому признаку. Воспользуемся признаками кратности произведения и суммы:

$$(a \cdot b)\ \%\ d = (a\ \%\ d)(b\ \%\ d)\ \%\ d$$
$$(a + b)\ \%\ d = (a\ \%\ d + b\ \%\ d)\ \%\ d$$

В первую очередь рассмотрим основание системы счисления и её степени

$$15\%14 = 1$$
$$15^2\%14 = (15\%14)(15\%14)\%\ 14 = 1 \cdot 1\ \%\ 14 = 1$$

Для остальных степеней будем иметь такой же остаток.

Теперь рассмотрим числа из условия $(9 \cdot 15^7 + 7 \cdot 15^6 + 9 \cdot 15^5 + 6 \cdot 15^4 + 8 \cdot 15^3 + x \cdot 15^2 + 1 \cdot 15^1 + 3 \cdot 15^0)\ \%\ 14$

$$9 \cdot 15^7\ \%\ 14 = (9\%14)(15^7\%14)\ \%\ 14 = (9 \cdot 1)\%\ 14 = 9$$

Аналогично рассматривая остальные разряды, получим, что признак кратности 14 числа в пятеричной системе равносилен признаку кратности 14 суммы цифр этого числа. То есть проверка на кратность 14 для нашего выражения будет равносильна проверке:

$$\big((9 + 7 + 9 + 6 + 8 + x + 1 + 3) + (7 + x + 2 + 1 + 3)\big)\%\ 14$$
$$\big((43 + x) + (13 + x)\big)\%\ 14$$
$$(56 + 2x)\%\ 14$$
$$(0 + 2x)\%\ 14$$
$$2x\ \%\ 14$$
$$\left[\begin{array}{l} x_1 = 0 \\ x_2 = 7 \\ x_3 = 14\ (E) \end{array}\right.$$

Следовательно, искомое значение x = 0.

$$+\ \begin{array}{r} 97968013 \\ 70213 \end{array}$$
$$= \overline{979D8226}$$

Переведем число в десятичную систему счисления и разделим на 14.

$$9 \cdot 15^7 + 7 \cdot 15^6 + 9 \cdot 15^5 + 13 \cdot 15^4 + 8 \cdot 15^3 + 2 \cdot 15^2 + 2 \cdot 15^1 + 6 \cdot 15^0 = 1624988736$$
$$1624988736 \div 14 = 116070624$$

Примечание: конечно же последние действия на экзамене нужно делать либо с помощью калькулятора, либо, например, с использованием функции *int(x, 15)* в языке программирования *python*. Например, так ***(int('97968013', 15) + int('70213', 15)) // 14***

Ответ: **116070624**

Решение (программное, переборное)

Python

```
# перебираем все цифры 14 системы счисления
# в порядке возрастания
for c in '0123456789acbde':
    # находим значение выражения
    x= int(f'97968{c}13',15)+int(f'7{c}213',15)
    # если число кратно 14
    if x % 14 == 0:
        # выводим частное
        print(x // 14)
        # заканчиваем поиск
        break
```

PascalABC.net

```
## uses school; // Автор: Максим Крючков
'0123456789ABCDE'
  .Select(
     x->Dec($'97968{x}13',15)+Dec($'7{x}213',15)
  )
  .Where(z -> z mod 14=0)
  .Select(z-> z/14).Min.print;
```

Ответ: **116070624**

15 Обозначим через m&n поразрядную конъюнкцию неотрицательных целых чисел m и n.
Так, например, $14\&5 = 1110_2\&0101_2 = 0100_2 = 4$.
Для какого наименьшего неотрицательного целого числа A формула

$$x\&39 = 0 \vee (x\&11 = 0 \rightarrow x\&A \neq 0)$$

тождественно истинна (т.е. принимает значение 1 при любом неотрицательном целом значении переменной x)?

Решение (аналитическое через анализ разрядов)
Переведем приведенные числа в двоичную систему счисления.
$39_{10} = 100111_2$ $11_{10} = 1011_2$
Проанализируем условие и определим, когда значение параметра влияет на значение всего выражения. Для удобства заменим подвыражения на коротки записи $\&_{39}$, $\&_{11}$ и $\&_A$.

$$\&_{39} \vee (\&_{11} \rightarrow \neg\&_A)$$

Значение подвыражения в скобке существенно, когда выражение $\&_{39} = 0$, значение подвыражения $\&_A$ влияет на значение подвыражения в скобке, когда $\&_{11} = 1$.
Итого имеем систему

$$\begin{cases}(x\&39 = 0) = 0\\(x\&11 = 0) = 1\\(x\&A \neq 0) = 1\end{cases}$$

$$\begin{cases}x\&39 \neq 0\\x\&11 = 0\\x\&A \neq 0\end{cases}$$

Таким образом, можно заключить, что в числе x биты на 0, 1 и 3 местах (считая с 0 справа) должны быть равны нулю. Также один из битов 0, 1, 2 или 5 должен быть равен 1. Или, соединяя эти два условия в одно, 0, 1 и 3 – нулевые биты, один из битов 2 или 5 равен 1.
Поэтому надо, чтобы параметр A покрыл варианты, когда или второй бит равен единице, или пятые бит равен единице, или оба бита (2 и 5) равны единице. Следовательно, минимальное значение параметра A, которое подходит под это ограничение, равно 36.

Ответ: **36**

Решение (аналитическое через алгебру логики)
Для удобства раскроем импликацию, сделав аналогичную предыдущему решению замену.

$$\&_{39} \vee (\&_{11} \rightarrow \neg\&_A) = \&_{39} \vee \neg\&_{11} \vee \neg\&_A$$

Опишем каждое подвыражение с помощью логических выражений, где X_n будет отвечать за значение разряда в позиции n числа x.

$$\&_{39} = (X_5{=}0) \wedge (X_2 = 0) \wedge (X_1 = 0) \wedge (X_0 = 0)$$
$$\neg\&_{11} = (X_3 = 1) \vee (X_1 = 1) \vee (X_0 = 1)$$

Можно заметить, что когда $X_n = 0$, значение подвыражения в $\&_{39}$ принимает значение истина, для второго наоборот. Поэтому для удобства можно заменить запись $(X_n = 0)$ на $\neg X_n$, $(X_n = 1)$ на X_n.

Подставим получившиеся выражения в выражение из условия и произведем сокращение выражения.

$$\neg X_5 \wedge \neg X_2 \wedge \neg X_1 \wedge \neg X_0 \vee X_3 \vee X_1 \vee X_0$$

По правилу свертки можем сократить выражение $\neg X_1 \wedge \neg X_0$. По итогу получим выражение

$$\neg X_5 \wedge \neg X_2 \vee X_3 \vee X_1 \vee X_0$$

Это выражение можно преобразовать (с учетом поиска минимальных значений) в выражение $\&_{36} \vee \&_8 \vee \&_2 \vee \&_1$
Дополним его подвыражением с параметром $\&_{36} \vee \neg\&_8 \vee \neg\&_2 \vee \neg\&_1 \vee \neg\&_A$
Заметим, что выражение может быть сведено к тожественной истинности только при сокращении его через правило исключения третьего для $\&_{36} \vee \neg\&_A$.

Ответ: **36**

Решение (программное)
Переберем все неотрицательные числа до 63 для x и A (64 ближайшая большая степень двойки для всех чисел в выражении) и найдем минимальное значение A, для которого выражение из задания истинно для всех перебираемых x.

Python

```
def f(x, A):
  return (x & 39 == 0) or (
            (x & 11 == 0) <= (x & A != 0))

for A in range(64):
  if all(f(x, A) for x in range(64)):
    print(A)
    break
```

PascalABC.net

```
## // Автор: Максим Крючков
(0..100).Where(A -> (0..1000)
  .All(
    x->(x and 39=0)or((x and 11=0)<=(x and A<>0)))
  )
  .Min.Print;
```

Ответ: **36**

16 Алгоритм вычисления значения функции *F(n)*, где n – натуральное число, задан следующими соотношениями:
F(n) = n, если n ≥ 2025,
F(n) =n + 3 + f(n+3), если n < 2025.
Чему равно значение выражения *F(23) – F(21)?*

Решение (аналитическое наблюдение)
Для выявления закономерности выражения *f(n) – f(n-2)* вычислим значения функции для чисел в диапазоне [2016; 2027].

2027	2026	2025	2024	2023	2022	2021	2020	2019	2018	2017	2016
2027	2026	2025	4054	4052	4050	6078	6075	6072	8099	8095	8091

Можно заметить, что разница между числом, кратным 3,и числом на 2 большим увеличивается с каждой тройкой на 2.
f(2027) – f(2025) = 2, f(2024) – f(2022) = 4, f(2021) – f(2019) = 6, f(2018) – f(2016) = 8. Следовательно, нужно понять, какой тройкой по счету будет тройка (23, 22, 21) и умножить её номер на 2.
(2027 – 23) / 3 + 1 = 2004/3 + 1 = 668 + 1 = 669

Ответ: **1338**
Решение (аналитическое, вывод формулы)
Заметим, что рекуррентная формула накапливает значение с шагом 3. То есть получает (*n*+3) +(*n*+6)+(*n*+9)+…+(*n*+3·*k*)+2·(*n*+3·(*k*+1)).
Где значение *k* – максимальное целое число, для которого выполняется неравенство *n* + 3·*k* < 2025.
Найдем значения *k* для 23 и 21.

$$23 + 3 \cdot k < 2025$$
$$3k < 2002$$
$$k < 667{,}33\ldots$$
$$k_{max} = 667$$

$$21 + 3 \cdot k < 2025$$
$$3k < 2004$$
$$k < 668$$
$$k_{max} = 667$$

Следовательно, имеем дело с двумя рядами одинаковой длины
f(23) = (*23*+3) +(*23*+6)+(*23*+9)+…+(*23*+3·*667*)+2·(*23*+3·(*667*+1))
f(21) = (*21*+3) +(*21*+6)+(*21*+9)+…+(*21*+3·*667*)+2·(*21*+3·(*667*+1))
Значит, при вычитании из f(23) значения f(21) получим набор разностей
f(23) – *f*(21) = ((*23*+3) - (*21*+3)) +((*23*+6)- (*21*+6))+((*23*+9)- (*23*+9))+…+((*23*+3·*667*)- (*21*+3·*667*))+2·((*23*+3·(*667*+1)))- 2·((*21*+3·(*667*+1)))

Или *f*(23) – *f*(21) = 2 + 2 + 2 +…+ 2 + 2·2 = 2·(667+2) = 1338

Ответ:**1338**

Решение (программное, рекурсия)
После беглого анализа можно понять, что больше 700 вложенных рекурсивных вызовов не будет. Следовательно, данный алгоритм можно переписать в код «буква в букву» и не ожидать переполнения стека.

Python
```
def f(n):
  if  n >= 2025:
    return n
  return n + 3 + f(n+3)
print(f(23) - f(21))
```

PascalABC.net
```
function f(n:integer):integer
  := (n>=2025 ? n : n+3+f(n+3));
print(f(23)-f(21));
```

Ответ: **1338**

Решение (программное, динамическое программирование)

Python
```
# заведем список для хранения значений
# последние значения определяются
# по условию выхода из рекурсии
f = [0]*2025 + [2025, 2026, 2027]
# перебираем значения в порядке убывания до 21
for n in range(2024, 20, -1):
  f[n] = n + 3 + f[n+3]
print(f[23] - f[21])
```

PascalABC.net

Ответ: **1338**

Решение (программное, динамическое программирование, короткий список)

Так как шаг рекурсии 3, то нет необходимости хранить все значения, достаточно хранить последние три вычисленных и обращаться к вычисленным ранее значениям по признаку остатка от деления на 3. Также разница между значениями 23 и 21 составляет 2, что означает, что списка из трех элементов точно хватит.

В списке для каждого значения будет сохраняться последнее вычисленное значение для соответствующего остатка. Таким образом, на каждой итерации будет перезаписываться элемент, который был вычислен 3 итерации назад.

Python	**PascalABC.net**

`f = [0]*3`
`for n in range(2025, 2028):`
`  f[n % 3] = n`
`# перебираем значения в порядке убывания до 21`
`for n in range(2024, 20, -1):`
`  f[n % 3] = n + 3 + f[n % 3]`
`print(f[23 % 3] - f[21 % 3])`	

Ответ: **1338**

Решение (электронные таблицы)

Автор: Эльвира Малышева

Для удобства в первый столбец заносим арифметическую прогрессию от 1 до 2030 (с небольшим запасом)

В столбце B для каждой ячейки заполняем формулу в соответствии с заданием B1=**ЕСЛИ(A1>=2025;A1;A1+3+B4)**

В отдельной ячейке считаем разность из задания F1=**B23-B21**

	A	B	C	D	E	F
1	1	687151				1338
2	2	687827				
3	3	686472				
4	4	687147				
5	5	687822				
6	6	686466				
7	7	687140				
8	8	687814				
9	9	686457				
10	10	687130				
11	11	687803				
12	12	686445				
13	13	687117				
14	14	687789				
15	15	686430				
16	16	687101				
17	17	687772				
18	18	686412				
19	19	687082				
20	20	687752				
21	21	686391				
22	22	687060				
23	23	687729				
24	24	686367				

Ответ: **1338**

17 ***Задание выполняется с использованием прилагаемых файлов.***

В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от 1 до 100 000 включительно. Определите количество пар последовательности, в которых только одно число трехзначное, и сумма элементов пары кратна минимальному трехзначному значению последовательности, оканчивающемуся на 5. В ответе запишите два числа: сначала количество найденных пар, затем **минимальную** из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности.

Решение (программное)

Python

```
# считываем последовательность чисел
nums = [int(x) for x in open('17.txt')]
# находим минимальное трехначное число
# оканчивающееся на 5
m5=min(x for x in num if 99<x<1000 and x%10==5)
pairs = []
# перебираем пары по условию
for a, b in zip(nums, nums[1: ]):
  if (99<a<1000) != (99<b<1000) \
     and (a+b) % m5 == 0:
    pairs.append(a + b)
print(len(pairs), min(pairs))
```

PascalABC.net

```
## // Автор: Максим Крючков
var t:=ReadAllText('17.txt').ToIntegers;
var min5 := t.Where(
    x-> (x mod 10=5) and (x in 100..999)
  ).Min;
var z:=t.Pairwise
  .Where(\(x,y)->
    ((x in 100..999) xor (y in 100..999))
    and (x+y).Divs(min5)
  )
  .Select(\(x,y) -> x+y);
Print(z.Count, z.Min);
```

Ответ: **2 33120**

Решение (электронные таблицы)

Откроем данные с помощью редактора электронных таблиц. Для нахождения минимального значения, кратного 105, воспользуемся следующим алгоритмом:

1) Напротив каждой ячейки с данными напишем формулу, которая либо возвращает число из данной строки, кратное 105, либо пустую строку
2) Среди полученных значений найдем минимальное

Формулы для первой строки:

1) =если(И(остат(A1;10)=5;A1>99;A1<1000); A1; "")

2) =макс(B:B)

Для выполнения пункта 2 в электронных таблицах, не поддерживающих столбчатые или строковые диапазоны, нужно указать первую и последнюю ячейки второго столбца (столбца B).

	A	B	C	D	E	F	G	H
1	567					115		
2	51360							
3	76956							
4	45215							
5	22129							
6	34648							
7	34651							
8	69023							
9	56338							
10	37358							
67	7318							
68	83474							
69	80200							
70	115	115						

Напишем формулу для определения трехзначного числа и преобразуем результат в 1 или 0.

=1*И(A1>99;A1<1000)

Начиная со второй строки, определим выполняемость условия на пару. Пара – число в предыдущей строке и число в текущей строке. Так как будем искать минимальное значение среди сумм квадратов в парах, в качестве неподходящего значения возьмем большое число. Адрес $F$1 абсолютный, так как в этой ячейке будет находится значение для формул во все строках столбца D.

=если(И(C1+C2=1;остат(A1+A2;$F$1)=0;A1+A2;"")

Теперь найдем ответы на вопросы задания.

Количество подходящих пар: **=счёт(D:D)**

Минимальная сумма значений элементов подходящих пар: **=мин(D:D)**

	A	B	C	D	E	F	G
1	567		1			115	
2	51360		0			2	
3	76956		0			33120	
4	45215		0				
5	22129		0				

Ответ: **2 33120**

18 ***Задание выполняется с использованием прилагаемых файлов.***

Квадрат разлинован на N × N клеток (1 < N < 30). Исполнитель Робот может перемещаться по клеткам, выполняя за одно перемещение одну из двух команд: вправо и вниз. По команде вправо Робот перемещается в соседнюю правую клетку, по команде вниз в соседнюю нижнюю. Квадрат ограничен внешними стенами. Между соседними клетками квадрата также могут быть внутренние стены. Сквозь стену Робот пройти не может. Перед каждым запуском Робота в каждой клетке квадрата лежит монета достоинством от 1 до 100. Посетив клетку, Робот забирает монету с собой; это также относится к начальной и конечной клеткам маршрута Робота. Определите максимальную и минимальную денежные суммы, которые может собрать Робот, пройдя из левой верхней клетки в правую нижнюю.

В ответе укажите два числа сначала максимальную сумму, затем минимальную. Исходные данные представляют собой электронную таблицу размером N × N, каждая ячейка которой соответствует клетке квадрата. Внутренние и внешние стены обозначены утолщенными линиями.

Пример входных данных

1	8	8	4
10	1	1	3
1	3	12	2
2	3	5	6

Для данных из примера ответ 34 22

Решение (электронные таблицы)

В первую очередь отформатируем таблицу таким образом, чтобы с ней было удобнее работать. Для этого клетки, в которые можно прийти только слева, покрасим в желтый цвет, клетки, в которые можно прийти только сверху, - в зеленый.

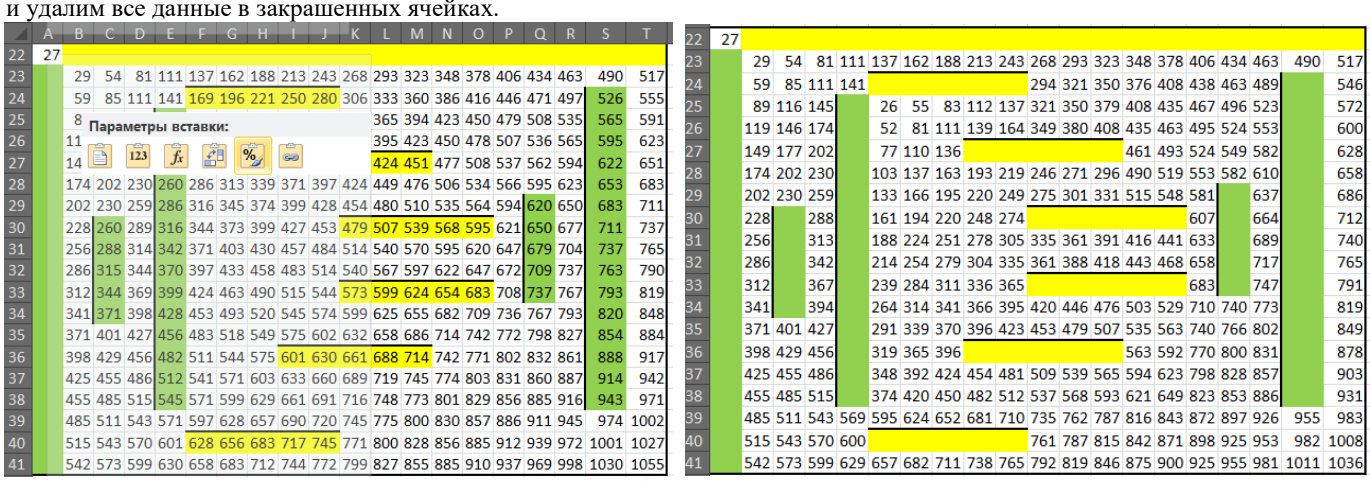
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	27	29	25	26	27	28	30	30	27	26	27	30	28	25	29	30	26	25	27	28
2	27	29	25	27	30	26	25	26	25	30	25	25	30	25	30	28	28	29	27	27
3	30	30	26	26	30	28	27	25	29	30	26	27	27	26	30	30	25	26	29	29
4	25	30	27	29	27	26	29	28	29	25	27	29	29	29	27	29	29	27	30	26
5	27	30	27	28	28	26	26	28	27	25	28	30	28	27	28	28	28	29	30	28
6	28	30	28	25	25	25	29	25	30	25	27	29	27	26	30	29	25	29	27	28
7	29	25	25	28	30	26	27	26	30	26	27	25	25	29	26	29	29	28	30	30
8	26	28	28	29	26	30	29	29	25	29	26	26	30	25	29	28	25	27	30	28
9	29	26	30	29	27	28	28	25	28	25	25	27	29	29	27	26	29	27	28	26
10	26	28	28	25	26	27	30	27	27	27	30	26	30	25	25	26	29	25	26	28
11	30	30	27	29	26	26	30	25	25	30	26	27	27	25	25	25	30	28	26	25
12	27	26	29	25	29	25	30	27	25	29	29	26	25	30	29	25	28	30	26	26
13	25	29	27	27	29	25	30	27	25	29	25	26	30	27	26	27	30	26	27	28
14	27	30	30	26	28	27	25	29	26	27	30	26	28	28	28	30	26	29	27	30
15	26	27	28	27	26	28	26	26	26	28	29	27	26	28	29	30	30	29	27	29
16	27	27	26	30	26	29	27	28	30	27	28	30	26	29	29	28	28	26	26	25
17	29	30	30	29	30	26	28	26	28	30	25	29	25	27	26	25	25	29	27	28
18	29	30	26	28	26	26	29	28	29	29	25	27	25	29	27	29	25	29	29	28
19	30	30	28	27	30	27	28	26	27	25	26	25	28	26	28	26	27	27	27	25
20	25	27	30	26	29	28	25	29	27	27	27	27	27	29	25	25	30	26	29	25

Производить расчеты будем в диапазоне A22:T41. В ячейку A22 перенесем значение из A1, например, с помощью формулы **=A1**.
После чего в ячейке B23 напишем формулу для выбора максимального счета в предыдущих ячейках (сверху и слева) и добавления к этому значению соответствующей ячейки из начального поля (B2) и заполним этой формулой весь диапазон B23:T41.

B23 **=макс(B22;A23)+B2**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
19	30	30	28	27	30	27	28	26	27	25	26	25	28	26	28	26	27	27	27	25
20	25	27	30	26	29	28	25	29	27	27	27	27	27	29	25	25	30	26	29	25
21																				
22	27																			
23		29	54	81	111	137	162	188	213	243	268	293	323	348	378	406	434	463	490	517
24		59	85	111	141	169	196	221	250	280	306	333	360	386	416	446	471	497	526	555
25		89	116	145	172	198	227	255	284	309	336	365	394	423	450	479	508	535	565	591
26		119	146	174	202	228	254	283	311	336	364	395	423	450	478	507	536	565	595	623
27		149	177	202	227	253	283	308	341	366	393	424	451	477	508	537	562	594	622	651
28		174	202	230	260	286	313	339	371	397	424	449	476	506	534	566	595	623	653	683
29		202	230	259	286	316	345	374	399	428	454	480	510	535	564	594	620	650	683	711
30		228	260	289	316	344	373	399	427	453	479	507	539	568	595	621	650	677	711	737
31		256	288	314	342	371	403	430	457	484	514	540	570	595	620	647	679	704	737	765
32		286	315	344	370	397	433	458	483	514	540	567	597	622	647	672	709	737	763	790
33		312	344	369	399	424	463	490	515	544	573	599	624	654	683	708	737	767	793	819
34		341	371	398	428	453	493	520	545	574	599	625	655	682	709	736	767	793	820	848
35		371	401	427	456	483	518	549	575	602	632	658	686	714	742	772	798	827	854	884
36		398	429	456	482	511	544	575	601	630	661	688	714	742	771	802	832	861	888	917
37		425	455	486	512	541	571	603	633	660	689	719	745	774	803	831	860	887	914	942
38		455	485	515	545	571	599	629	661	691	716	748	773	801	829	856	885	916	943	971
39		485	511	543	571	597	628	657	690	720	745	775	800	830	857	886	911	945	974	1002
40		515	543	570	601	628	656	683	717	745	771	800	828	856	885	912	939	972	1001	1027
41		542	573	599	630	658	683	712	744	772	799	827	855	885	910	937	969	998	1030	1055

Теперь скопируем форматирование исходного поля (ПКМ-Копировать или Ctrl+C) и вставим только форматирование на заполняемое поле и удалим все данные в закрашенных ячейках.

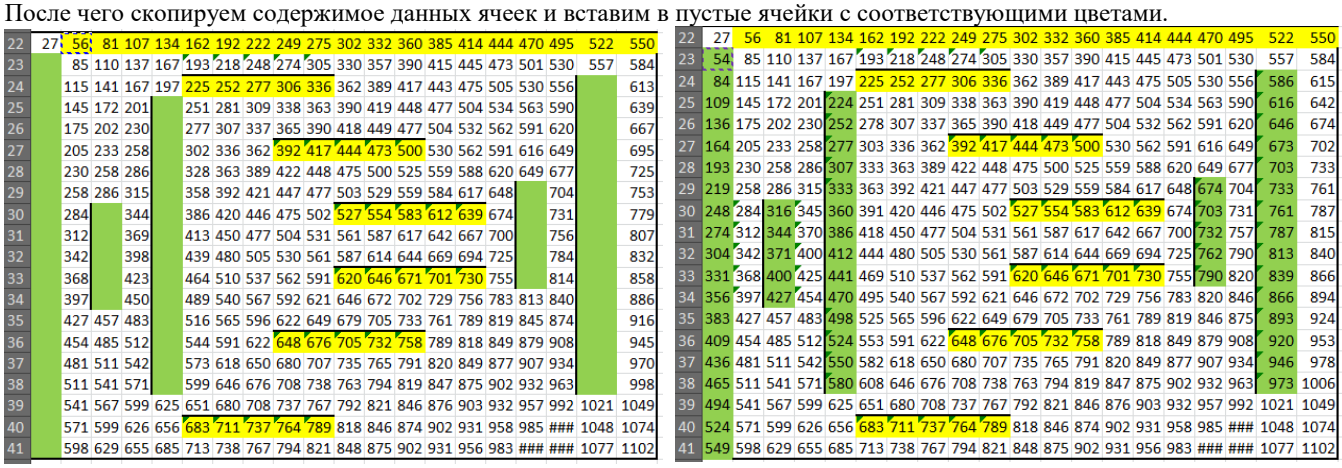


И заполним ячейки B22 и A23 соответствующими для них формулами (только движение слева и только движение справа)

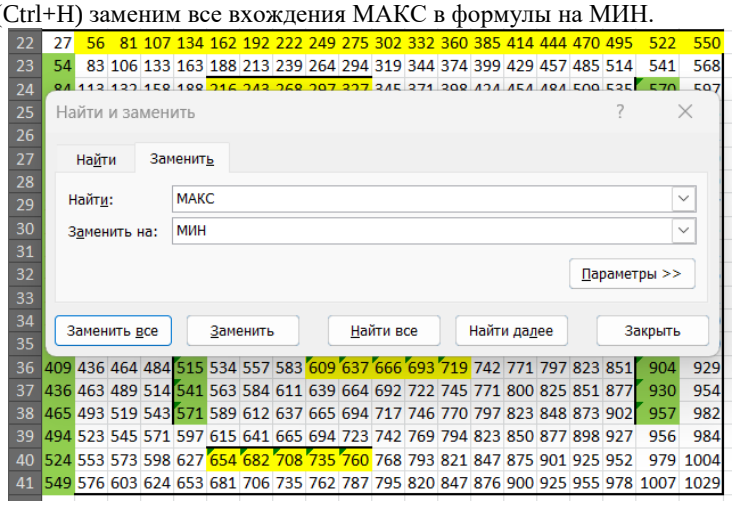
B22=**A22+B1**

A23=**A22+A2**

После чего скопируем содержимое данных ячеек и вставим в пустые ячейки с соответствующими цветами.



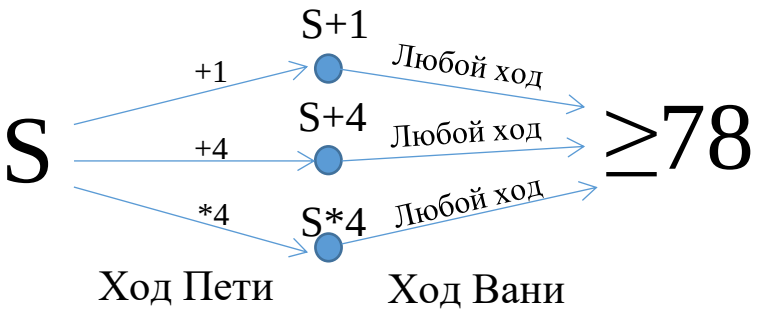
Теперь с помощью меню замены (Ctrl+H) заменим все вхождения МАКС в формулы на МИН.



Ответ: **1102 1029**

19 Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один** или **четыре** камня, либо увеличить количество камней в куче **в четыре раза**. У каждого игрока есть неограниченное количество камней, чтобы делать ходы.
Игра завершается в тот момент, когда количество камней в куче становится не менее 78.
Победителем **считается** игрок, сделавший последний ход, т.е. первым получивший кучу из 78 или более камня.
В начальный момент в куче было S камней; $1 \le S \le 77$.
Будем говорить, что игрок имеет выигрышную стратегию, если он может выиграть при любых ходах противника.
Укажите **минимальное** значение S, при котором Ваня может выиграть своим первым ходом после любого хода Пети.

Решение (аналитическое)
Изобразим игру схематически.



Сперва определим все значения S, при которых ходящий из них игрок может выиграть первым ходом (одним или несколькими из допустимых ходов).

$$\begin{bmatrix} \text{ход} + 1: S = 77 \\ \text{ход} + 4: 74 \le S \le 77 \\ \text{ход} * 4: 20 \le S \le 77 \end{bmatrix}$$

Следовательно, при значениях S, принадлежащих диапазону [20; 77], игрок, делающий ход из них одерживает победу своим первым ходом (выигрышные позиции для игры длиной в один ход).
Чтобы найти выигрышные позиции для игры длиной в два хода (вопрос задачи), необходимо, чтобы после первого хода следующий игрок ходил из выигрышной позиции для игры длиной в один ход.

$$\begin{cases} 20 \le S + 1 \le 77 \\ 20 \le S + 4 \le 77 \\ 20 \le 4S \le 77 \\ S \notin [20;\ 77] \end{cases} \quad \begin{cases} 19 \le S \le 76 \\ 16 \le S \le 73 \\ 5 \le S \le 19 \\ S \notin [20;\ 77] \end{cases} \quad S = 19$$

Ответ: **19**

Решение (электронные таблицы, формула)
Вставим в столбец А с ячейки А1 числа арифметической прогрессии от 1 до 78 с шагом 1.
В ячейке В1 запишем формулу для победы первым ходом В1 =ЕСЛИ(ИЛИ(A1+1>79;A1+4>79;A1*4>79);1;0)
В ячейке С1 запишем формулу для победы вторым ходом
С1 =ЕСЛИ(B1=1;0;ЕСЛИ(И(B2=1;B5=1;ИНДЕКС(B:B;$A1*4)=1);2;0))
Условие В1=1 гарантирует, что проверяемое значение не удовлетворяло игре в один ход. ИНДЕКС(B:B;A1*4) возвращает значение для 4S в столбце В.Также для удобства поиска найденных значений наложим на диапазон B:E условное форматирование для значений неравных 0.

	A	B	C	D	E	F	G	H
1	1	0	0					
2	2	0	0					
3	3	0	0					
4	4	0	0					
5	5	0	0					
6	6	0	0					
7	7	0	0					
8	8	0	0					
9	9	0	0					
10	10	0	0					
11	11	0	0					
12	12	0	0					
13	13	0	0					
14	14	0	0					
15	15	0	0					
16	16	0	0					
17	17	0	0					
18	18	0	0					
19	19	0	2					
20	20	1	0					
21	21	1	0					
22	22	1	0					

Решение (программное, переборное)

Перенесем в код вопрос задачи.

Python	PascalABC.net

`for S in range(1, 77):`
`  if all(s1 < 78`
`  and any(s2 >= 78`
`  for s2 in (s1+1, s1+4, s1*4))`
`  for s1 in (S+1, S+4, S*4)):`
`  print(S)`
`  break`	

Решение (программное, на множествах)

Построено на поэтапном нахождении позиций. Сначала в множестве s1 найдем значения S для позиций, при ходе из которых можно сразу выиграть (игра длиной 1). Затем найдем все значения S для позиций, при ходе из которых игрок может попасть только в позиции для игры длиной 1 (множество s1)

Python	PascalABC.net

`# множество вершин для игры в 1 ход`
`s1 = set()`
`for s in range(1, 77):`
`  if any(x >= 78 for x in (s+1, s+4, s*4)):`
`  s1.add(s)`
`# множество вершин для игры в 2 хода`
`s2 = set()`
`for s in range(1, 77):`
`  if all(x in s1 for x in (s+1, s+4, s*4)):`
`  s2.add(s)`
`print(min(s2))`
	`## // Автор: Максим Крючков`
	`// Множество еще не исследованных позиций`
	`var kuk := (1..77).ToHashSet;`
	`// Множество вершин для игр в один ход`
	`var Win1 := kuk.Where(`
	`  k-> |k+1,k+4,k*4|.Any(t-> t>=78))`
	`  .ToHashSet;`
	`// Множество вершин для игры в два хода`
	`var Los1 := kuk.Where(`
	`  k-> |k+1,k+4,k*4|.All(t->t in Win1))`
	`  .ToHashSet.Println;`

Решение (программное, на списке)

Заведем список *pos* из 77 нулей (78 с учетом нулевой ячейки). В качестве значений *pos[S]* в список запишем 1 для тех значений S, для которых игра может быть завершена за один ход. Затем запишем 2 в те элементы *pos[S]*, из которых любой ход приходит в значения S из которых можно выиграть первым ходом (*pos[S+1] = pos[S+4] = pos[S*4]* = 1).

Python	PascalABC.net

`pos = [0]*78`
`for S in range(1, 78):`
`  if any(s1 >= 78 for s1 in (S+1, S+4, S*4)):`
`  pos[S] = 1`
`for S in range(1, 78):`
`  if pos[S] == 0 \`
`  and all(pos[s2]==1 for s2 in (S+1,S+4,S*4)):`
`  pos[S] = 2`
 	
`print(min(S for S in range(1,78) if pos[S]==2))`	

Решение (рекурсивная функция, возвращающая длину игры)

Логика работы:

- если можно завершить игру победой в один ход, функция возвращает 1 (игра длиной 1).
- если из всех позиций, в которые можно попасть из S, есть возможность завершить игру в 1 ход, вернуть 2 (игра длиной 2).

Python	PascalABC.net

`def game(S):`
`  if any(s1 >= 78 for s1 in (S+1, S+4, S*4)):`
`  return 1`
`  if all(game(s2)==1 for s2 in (S+1,S+4,S*4)):`
`  return 2`
 	
`print(min(S for S in range(1,78)`
`  if game(S)==2))`	

Для удобства интерпретации можно заменить 1 на строку "P1", 2 – на "V1". Тогда будет более нагляден факт победы Пети одним ходом или Вани одним ходом.

Решение (рекурсивная функция, окончание игры за N-2k ходов)

Наличие стратегии у игрока показывает, что игра может закончиться за нечетное число шагов, если выигрывает первый игрок, и за четное количество шагов, если выигрывает второй игрок. Таким образом, функция будет работать с параметрами: S – исследуемое значение, count – количество шагов, за которое нужно точно победить, step – сколько шагов уже было сделано.

Например, вызов функции game(15, 4, 1) вернет истину в том случае, если игра завершится победой второго игрока не более, чем за два его хода, с учетом того, что до вершины для значения S уже был совершен один ход.

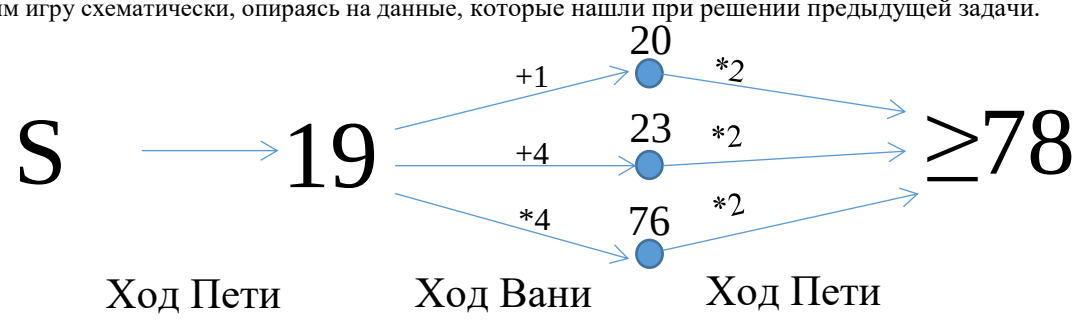
Python	PascalABC.net

```
def game(S, count, step):
  if S >= 78:
    # ход игрока, который должен завершить
    if count == step:
      return 1
    else:
      return 0
  # если шаги закончились, а игра нет
  if count <= step:
    return 0
  # первый ход для игры длиной 2
  if count == 2 and step == 0:
    # все партии должны вернуть истину
    return all(game(s1, count, 1)
               for s1 in (S+1, S+4, S*4))
  # второй ход для игры длиной 2
  if count == 2 and step == 1:
    # хотя бы одна партия должна быть победной
    return any(game(s1, count, 2)
               for s1 in (S+1, S+4, S*4))
  # первый ход для игры длиной 1
  if count == 1 and step == 0:
    # хотя бы один ход приводит к победе
    return any(game(s1, count, 1)
               for s1 in (S+1, S+4, S*4))
print(min(S for S in range(1,78)
          if game(S,2,0)))
```

20 Для игры, описанной в задании 19, **найдите два минимальных значения S**, при которых у Пети есть выигрышная стратегия вторым ходом, при этом он не может гарантировано выиграть за один ход.

Решение (аналитическое)

Изобразим игру схематически, опираясь на данные, которые нашли при решении предыдущей задачи.



$$\begin{cases} S+1=19 \\ S+4=19 \\ 4S=19 \end{cases}$$

Решим совокупность в целых числах

$$\begin{cases} S=18 \\ S=15 \end{cases}$$

Ответ: **15 18**

Решение (электронные таблицы, формула)

В ячейке D1 запишем формулу =ЕСЛИ(СУММ($B1:C1)<>0;0;ЕСЛИ(ИЛИ(C2=2;C5=2;ИНДЕКС(C:C;$A1*4)=2);3;0))

Условие СУММ($B1:C1)<>0 означает, что для предыдущих ходов не находилось значений для игр длиной 1 или 2.

	A	B	C	D	E	F	G
13	13	0	0	0			
14	14	0	0	0			
15	15	0	0	3			
16	16	0	0	0			
17	17	0	0	0			
18	18	0	0	3			
19	19	0	2	0			
20	20	1	0	0			
21	21	1	0	0			
22	22	1	0	0			
23	23	1	0	0			

Решение (программное, на множествах)

Python

```
# множество вершин для игры в 1 ход
s1 = set()
for s in range(1, 77):
  if any(x >= 78 for x in (s+1, s+4, s*4)):
    s1.add(s)
# множество вершин для игры в 2 хода
s2 = set()
for s in range(1, 77):
  if all(x in s1 for x in (s+1, s+4, s*4)):
    s2.add(s)
# множество вершин для игры в 3 хода
s3 = set()
for s in range(1, 77):
  if all(x in s2 for x in (s+1, s+4, s*4)):
    s3.add(s)

print(sorted(s3)[:2])
```

PascalABC.net

```
## // Автор: Максим Крючков
// Множество еще не исследованных позиций
var kuk := (1..77).ToHashSet;
// Множество вершин для игр в один ход
var Win1 := kuk.Where(
              k-> |k+1,k+4,k*4|.Any(t-> t>=78))
            .ToHashSet;
//Удаляем найденные позиции из дальнейшего анализа
kuk -= Win1;
// Множество вершин для игры в два хода
var Los1 := kuk.Where(
              k-> |k+1,k+4,k*4|.All(t->t in Win1))
            .ToHashSet;
kuk -= Los1; // так же удаляем
// Множество вершин для игры в три хода
var Win2 := kuk.Where(
              k-> |k+1,k+4,k*4|.Any(t->t in Los1))
            .ToHashSet.Println;
```

Решение (программное, на списке)

Python

```
pos = [0]*78
for S in range(1, 78):
  if any(s1 >= 78 for s1 in (S+1, S+4, S*4)):
    pos[S] = 1
for S in range(1, 78):
  if pos[S] == 0 \
   and all(pos[s2]==1 for s2 in (S+1,S+4,S*4)):
    pos[S] = 2
for S in range(1, 78):
  if pos[S] == 0 \
   and any(pos[s3]==2 for s3 in (S+1,S+4,S*4)):
    pos[S] = 3

print([S for S in range(1,78)
       if pos[S]==3][:2])
```

PascalABC.net

Решение (рекурсивная функция, возвращающая длину игры)

Python

```
def game(S):
  if any(s1 >= 78 for s1 in (S+1, S+4, S*4)):
    return 1
  if all(game(s2)==1 for s2 in (S+1,S+4,S*4)):
    return 2
  if any(game(s3)==2 for s3 in (S+1,S+4,S*4)):
    return 3

print([S for S in range(1,78)
       if game(S)==3][:2])
```

PascalABC.net

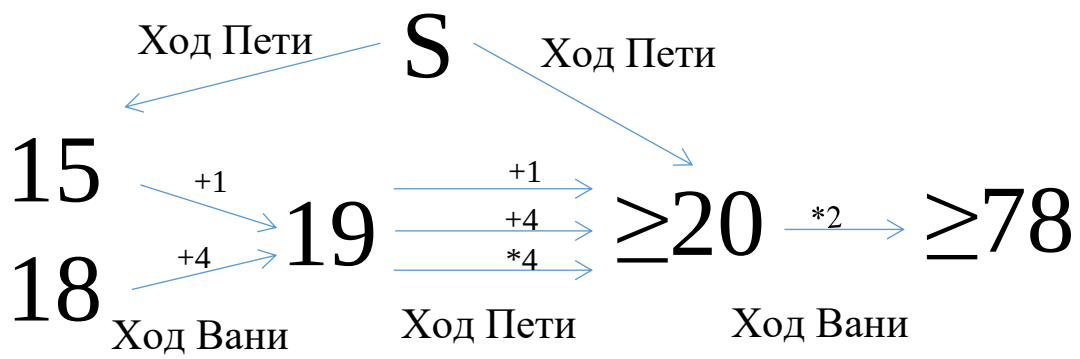
Решение (рекурсивная функция, окончание игры за N-2k ходов)
Заметим, что когда разница между count и step четная, то проверка идет для проигрывающего игрока. Поэтому нужно проверить все следующие ходы на «проигрыш», иначе нужно, чтобы хотя бы одна игра была победной (для игры целевого игрока).

Python	PascalABC.net

```
def game(S, count, step):
  if S >= 78:
    return count == step
  # если шаги закончились, а игра нет
  if count <= step:
    return 0
  # если ход проигрывающего игрока
  if (count - step) % 2 == 0:
    return all(game(s1, count, step+1)
               for s1 in (S+1, S+4, S*4))
  else:
    return any(game(s1, count, step+1)
               for s1 in (S+1, S+4, S*4))
print([S for S in range(1,78)
         if game(S,3,0)][:2])
```

21 Для игры, описанной в задании 19, найдите **минимальное** значение S, при котором одновременно выполняются два условия:
— у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
— у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Решение (аналитическое)
Изобразим игру схематически, опираясь на данные, которые нашли при решении предыдущей задачи.

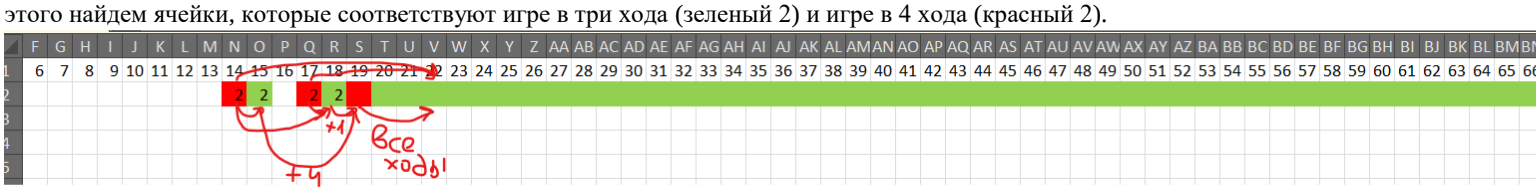


То есть, необходимо, чтобы ход из позиции для S был либо в значения 15 или 18, или в позицию с количеством камней от 20 до 77. При этом значение S не должно входить в уже найденные множества для задач 19 и 20.

$$\left\{\begin{array}{l}\left[\begin{array}{l}S+1=15\\S+1=18\\S+1\geq 20\end{array}\right.\\\left[\begin{array}{l}S+4=15\\S+4=18\\S+4\geq 20\end{array}\right.\\\left[\begin{array}{l}S*4=15\\S*4=18\\S*4\geq 20\end{array}\right.\end{array}\right.\quad\left\{\begin{array}{l}\left[\begin{array}{l}S=14\\S=17\\S\geq 19\end{array}\right.\\\left[\begin{array}{l}S=11\\S=14\\S\geq 16\end{array}\right.\\\left[S\geq 5\right.\end{array}\right.\quad\left[\begin{array}{l}S=14\\S=17\end{array}\right.$$

Ответ:**14**

Решение (ручное, эл.таблицы)
В первой строке укажем количество камней в куче.
Во второй строке последовательно закрасим сначала клетки, соответствующие игре в один ход (зеленый), затем в два (красный). После этого найдем ячейки, которые соответствуют игре в три хода (зеленый 2) и игре в 4 хода (красный 2).



Решение (электронные таблицы, формула)

Формула E1 =ЕСЛИ(СУММ($B1:D1)<>0;0;
ЕСЛИ(И(ИЛИ(D2=3;B2=1);ИЛИ(D5=3;B5=1);ИЛИ(ИНДЕКС(D:D;$A1*4)=3;ИНДЕКС(B:B;$A1*4)=1));4;0))

В этой формуле меняем проверку предыдущих ходов, чтобы текущий ход приходил либо в 1 или в 3.

	A	B	C	D	E	F	G
10	10	0	0	0	0		
11	11	0	0	0	0		
12	12	0	0	0	0		
13	13	0	0	0	0		
14	14	0	0	0	4		
15	15	0	0	3	0		
16	16	0	0	0	0		
17	17	0	0	0	4		
18	18	0	0	3	0		
19	19	0	2	0	0		
20	20	1	0	0	0		

Решение (программное, на множествах)

Так как мы будем находить победу вторым или четвертым ходом, необходимо ограничить перебор значений условием, что перебираемое значение не входит в множество *s2*.

Python

```
# множество вершин для игры в 1 ход
s1 = set()
for s in range(1, 77):
  if any(x >= 78 for x in (s+1, s+4, s*4)):
    s1.add(s)
# множество вершин для игры в 2 хода
s2 = set()
for s in range(1, 77):
  if all(x in s1 for x in (s+1, s+4, s*4)):
    s2.add(s)
# множество вершин для игры в 3 хода
s3 = set()
for s in range(1, 77):
  if all(x in s2 for x in (s+1, s+4, s*4)):
    s3.add(s)
# множество вершин для четвертого хода
s4 = set()
for s in range(1, 77):
  if s not in s2 and\
    all(x in s1|s3 for x in (s+1, s+4, s*4)):
    s4.add(s)

print(min(s4))
```

PascalABC.net

```
## // Автор: Максим Крючков
// Множество еще не исследованных позиций
var kuk := (1..77).ToHashSet;
// Множество вершин для игр в один ход
var Win1 := kuk.Where(
              k-> |k+1,k+4,k*4|.Any(t-> t>=78))
            .ToHashSet;
//Удаляем найденные позиции из дальнейшего анализа
kuk -= Win1;
// Множество вершин для игры в два хода
var Los1 := kuk.Where(
              k-> |k+1,k+4,k*4|.All(t->t in Win1))
            .ToHashSet;
kuk -= Los1; // так же удаляем
// Множество вершин для игры в три хода
var Win2 := kuk.Where(
              k-> |k+1,k+4,k*4|.Any(t->t in Los1))
            .ToHashSet;
kuk-=Win2; // так же удаляем
// Множество вершин для игры в четыре хода
var Los2 := kuk.Where(
              k-> |k+1,k+4,k*4|
                    .All(t->t in Win1+Win2))
            .ToHashSet.Min.Println;
```

Решение (программное, на списке)

Python

```
pos = [0]*78
for S in range(1, 78):
  if any(s1 >= 78 for s1 in (S+1, S+4, S*4)):
    pos[S] = 1
for S in range(1, 78):
  if pos[S] == 0 \
   and all(pos[s2]==1 for s2 in (S+1,S+4,S*4)):
    pos[S] = 2
for S in range(1, 78):
  if pos[S] == 0 \
   and any(pos[s3]==2 for s3 in (S+1,S+4,S*4)):
    pos[S] = 3
for S in range(1, 78):
  if pos[S] == 0 \
   and all(pos[s4] in (1, 3)
           for s4 in (S+1,S+4,S*4)):
    pos[S] = 4

print([S for S in range(1,78) if pos[S]==4][0])
```

PascalABC.net

Решение (рекурсивная функция, возвращающая длину игры)

Однако, при таком подходе мы столкнемся с долгим выполнением программы. Дело в том, что для небольших значений мы будем очень медленно подниматься к условию выхода из рекурсии. Чтобы ускорить процесс обхода значений упорядочим возможные значения S для следующего шага в порядке убывания. Так алгоритм будет быстрее доходить до условия выхода из рекурсии.

Python	PascalABC.net

`def game(S):`
`  if any(s1 >= 78 for s1 in (S*4, S+4, S+1)):`
`  return 1`
`  if all(game(s2)==1 for s2 in (S*4,S+4,S+1)):`
`  return 2`
`  if any(game(s3)==2 for s3 in (S*4,S+4,S+1)):`
`  return 3`
`  if all(game(s4) in (1, 3)`
`  for s4 in (S*4,S+4,S+1)):`
`  return 4`
 	
`print([S for S in range(1,78)if game(S)==4][0])`	

Решение (рекурсивная функция, окончание игры за N-2k ходов)

Проанализировав схему игры для четырех ходов, понимаем, что искомая стратегия может содержать одну или несколько ветвей с двумя ходами. Поэтому нам необходимо заменить условие победы на проверку признака четности *count* и *step*.

Python	PascalABC.net

`def game(S, count, step):`
`  if S >= 78:`
`  return count % 2 == step % 2`
`  # если шаги закончились, а игра нет`
`  if count <= step:`
`  return 0`
`  # если ход проигрывающего игрока`
`  if (count - step) % 2 == 0:`
`  return all(game(s1, count, step+1)`
`  for s1 in (S+1, S+4, S*4))`
`  else:`
`  return any(game(s1, count, step+1)`
`  for s1 in (S+1, S+4, S*4))`
`print([S for S in range(1,78)`
`  if game(S,4,0)][0])`	

Примечание: в конкретном решении нет необходимости учитывать, что при таком изменении функции некоторые значения, возвращаемые для 4 ходов, будут возвращаться и функцией, которая находит значения для 2 ходов. Более корректным вариантом описания условия для 4 ходов было бы выражение ***not game(S, 2, 0) and game(S, 4, 0)***.

22 **Задание выполняется с использованием прилагаемых файлов.**

В файле содержится информация о совокупности N вычислительных процессов, которые могут выполняться параллельно или последовательно. Будем говорить, что процесс B зависит от процесса A, если для выполнения процесса B необходимы результаты выполнения процесса A. В этом случае процессы могут выполняться только последовательно.
Информация о процессах представлена в файле в виде таблицы. В первой строке таблицы указан идентификатор процесса (ID), во второй строке таблицы – время его выполнения в миллисекундах, в третьей строке перечислены с разделителем «;» ID процессов, от которых зависит данный процесс. Если процесс является независимым, то в таблице указано значение 0.

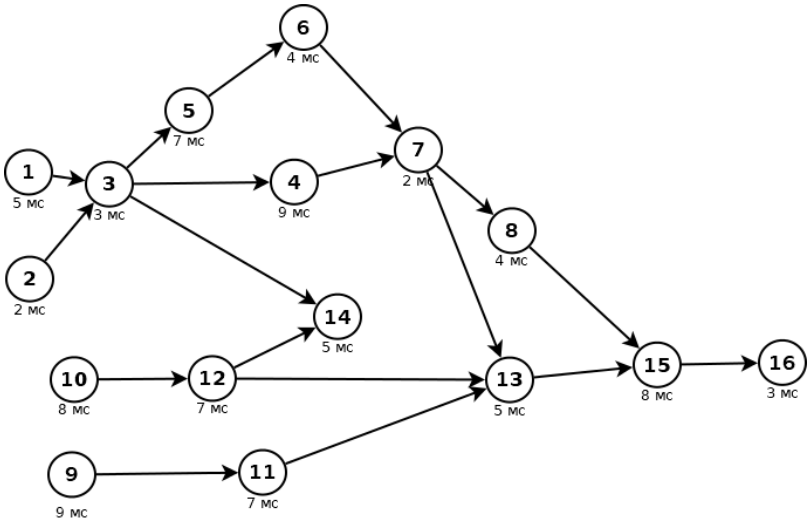
ID процесса B	*Время выполнения процесса B (мс)*	*ID процесса(-ов) A*
1	4	0
2	3	0
3	1	1; 2
4	7	3

Определите минимальное время, через которое завершится выполнение всей совокупности процессов, при условии, что все независимые друг от друга процессы могут выполняться параллельно.

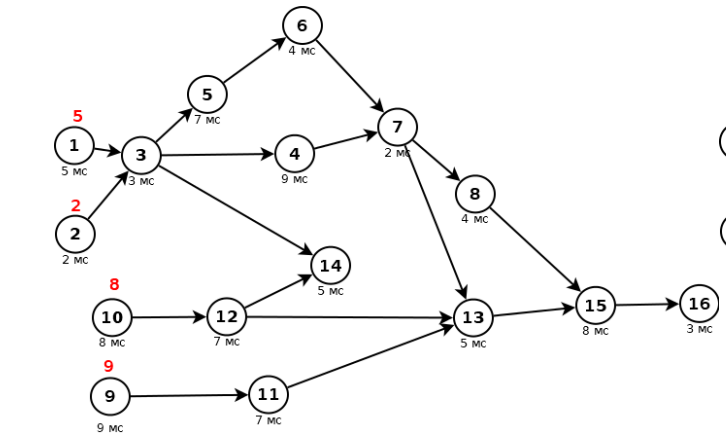
Типовой пример имеет иллюстративный характер. Для выполнения задания используйте данные из прилагаемого файла.

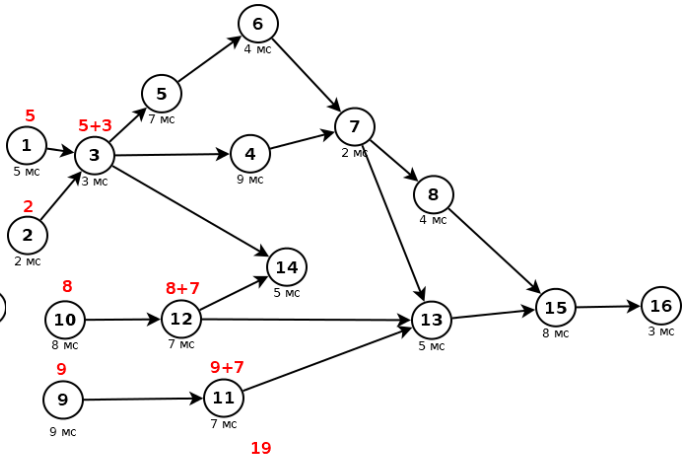
Решение (графическое)

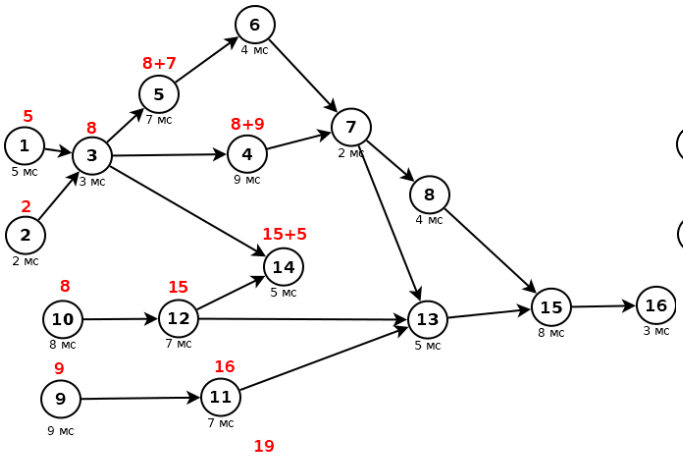
	A	B	C	D
1	ID процес	Время выполнени	ID процессов A	
2	1	5	0	
3	2	2	0	
4	3	3	1; 2	
5	4	9	3	
6	5	7	3	
7	6	4	5	
8	7	2	4; 6	
9	8	4	7	
10	9	9	0	
11	10	8	0	
12	11	7	9	
13	12	7	10	
14	13	5	7; 11; 12	
15	14	5	3; 12	
16	15	8	8;13	
17	16	3	15	

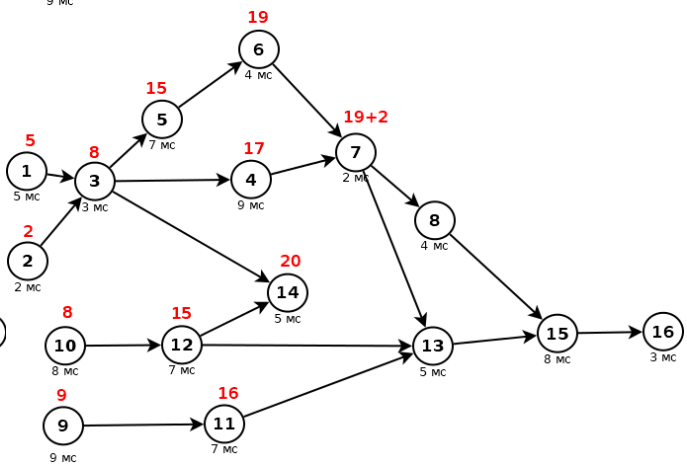


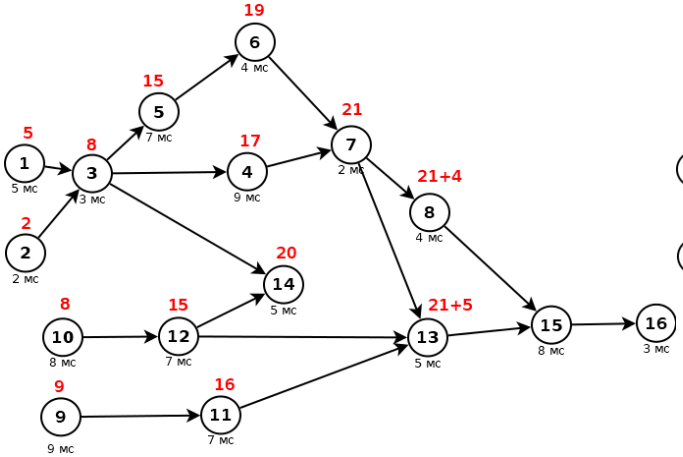
Теперь расставим значения для начальных процессов над ними и к максимальному значению для каждого предшествующего процесса будем добавлять время текущего. После чего выберем максимальное полученное значение среди конечных вершин (14 и 16 процессы).

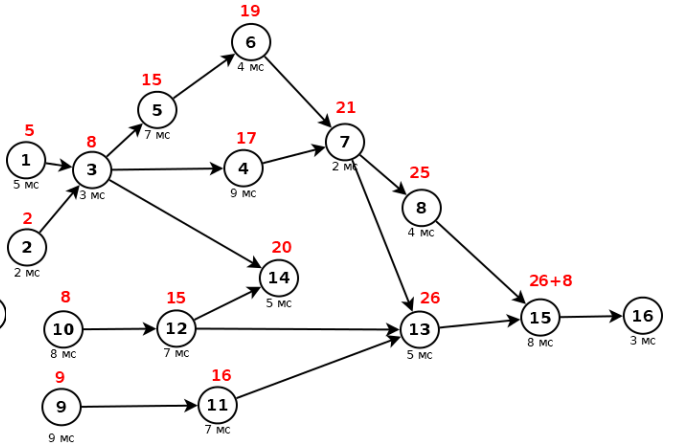


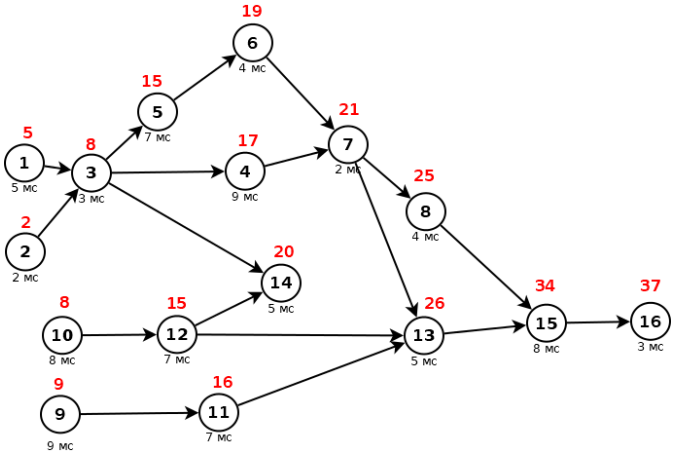












Ответ: **37**

Решение (ручное, диаграмма Ганта)

Справа от процессов создадим временную шкалу, где одна ячейка будет соответствовать одной миллисекунде. Зависимые процессы будем начинать на одну клетку правее всех процессов, от которых они зависят.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS
1	ID процес	Время в	ID процессо		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
2	1	5	0																																											
3	2	2	0																																											
4	3	3	1; 2																																										
5	4	9	3																																										
6	5	7	3																																										
7	6	4	5																																										
8	7	2	4; 6																																										
9	8	4	7																																										
10	9	9	0																																										
11	10	8	0																																										
12	11	7	9																																										
13	12	7	10																																										
14	13	5	7; 11; 12																																										
15	14	5	3; 12																																										
16	15	8	8;13																																										
17	16	3	15																																										

Заметим, что последним завершается процесс №16 на отметке 37 мс.

Ответ: **37**

Решение (полуручное, таблица)

Заведем в таблице еще один столбец, в котором будем определять минимальное время завершения для каждого процесса. В качестве начальных данных занесем в таблицу время выполнения всех независимых процессов. Далее значения будем определять по формуле =МАКС(список ячеек) + время выполнения процесса. Где в список ячеек будут входить соответствующие списку процессов, от которых зависит процесс, для которого мы подсчитываем время выполнения. Для процессов, которые зависят только от одного процесса, можно не использовать функцию МАКС, например, для процессов 3 и 4 можно написать такие формулы.

	A	B	C	D
1	ID процес	Время в	ID процессов А	Время завершения
2	1	5	0	5
3	2	2	0	2
4	3	3	1; 2	
5	4	9	3	
6	5	7	3	
7	6	4	5	
8	7	2	4; 6	
9	8	4	7	
10	9	9	0	9
11	10	8	0	8
12	11	7	9	
13	12	7	10	
14	13	5	7; 11; 12	
15	14	5	3; 12	
16	15	8	8;13	
17	16	3	15	

	A	B	C	D	E
1	ID процес	Время в	ID процессов А	Время завершения	
2	1	5	0	5	
3	2	2	0	2	
4	3	3	1; 2	=макс(D2;D3)+B4	
5	4	9	3		
6	5	7	3		

	A	B	C	D	E
1	ID процес	Время в	ID процессов А	Время завершения	
2	1	5	0	5	
3	2	2	0	2	
4	3	3	1; 2	8	
5	4	9	3	=D4+B5	

Заполним формулы для всех ячеек и найдем максимальное значение с помощью формулы D18=**МАКС(D2:D17)**

	A	B	C	D
1	ID процес	Время в	ID процессов А	Время завершения
2	1	5	0	5
3	2	2	0	2
4	3	3	1; 2	8
5	4	9	3	17
6	5	7	3	15
7	6	4	5	19
8	7	2	4; 6	21
9	8	4	7	25
10	9	9	0	9
11	10	8	0	8
12	11	7	9	16
13	12	7	10	15
14	13	5	7; 11; 12	26
15	14	5	3; 12	20
16	15	8	8;13	34
17	16	3	15	37
18				**37**

Решение (через замену)

Сформируем формулы из предыдущего способа решения через меню замены. Для того, чтобы номера строк соответствовали номерам процессов удалим строку с заголовком таблицы. Также для удобства замены удалим все пробелы, заменив (Ctrl+H) пробелы на пустые последовательности.

Теперь нам необходимо сформировать формулу вида МАКС(ячейка1;ячейка2). Так как работать мы будем в столбце D, соответственно для нескольких ячеек необходимо будет разделить адреса ячеек по точке с запятой. Поэтому произведем замену точки с запятой «;» на точку с запятой и букву D «;D».

После первых замен получим такие результаты

	A	B	C	D
1	1	5	0	
2	2	2	0	
3	3	3	1;2	
4	4	9	3	
5	5	7	3	
6	6	4	5	
7	7	2	4;6	
8	8	4	7	
9	9	9	0	
10	10	8	0	
11	11	7	9	
12	12	7	10	
13	13	5	7;11;12	
14	14	5	3;12	
15	15	8	8;13	
16	16	3	15	

	A	B	C
1	1	5	0
2	2	2	0
3	3	3	1;D2
4	4	9	3
5	5	7	3
6	6	4	5
7	7	2	4;D6
8	8	4	7
9	9	9	0
10	10	8	0
11	11	7	9
12	12	7	10
13	13	5	7;D11;D12
14	14	5	3;D12
15	15	8	8;D13
16	16	3	15

Теперь из полученных строк достаточно просто сформировать необходимую формулу. Однако, надо помнить, что для независимых процессов необходимо определить время выполнения, равное времени выполнению данных процессов. Поэтому используем условную функцию.

D1=**ЕСЛИ(C1=0;B1; "МАКС(D"&C1&")+"&B1)**

После чего скопируем значения, которые получились в результате работы формулы и вставим только значения в наш диапазон (скопируем ячейки D1:D16 и вставим их в этот же диапазон по значению). После чего заменим все вхождения «МАКС» на «=МАКС». Важно сразу не делать замену в строке на «=МАКС», тогда нужно будет вручную прокликивать все ячейки, так как при вставке значений вставляются строковые значения, которые автоматически не преобразуются в формулы (соответственно, не высчитываются)

	A	B	C	D
1	1	5	0	5
2	2	2	0	2
3	3	3	1;D2	МАКС(D1;D2)+3
4	4	9	3	МАКС(D3)+9
5	5	7	3	МАКС(D3)+7
6	6	4	5	МАКС(D5)+4
7	7	2	4;D6	МАКС(D4;D6)+2
8	8	4	7	МАКС(D7)+4
9	9	9	0	9
10	10	8	0	8
11	11	7	9	МАКС(D9)+7
12	12	7	10	МАКС(D10)+7
13	13	5	7;D11;D12	МАКС(D7;D11;D12)+5
14	14	5	3;D12	МАКС(D3;D12)+5
15	15	8	8;D13	МАКС(D8;D13)+8
16	16	3	15	МАКС(D15)+3

Параметры вставки:

	A	B	C	D
1	1	5	0	5
2	2	2	0	2
3	3	3	1;D2	8
4	4	9	3	17
5	5	7	3	15
6	6	4	5	19
7	7	2	4;D6	21
8	8	4	7	25
9	9	9	0	9
10	10	8	0	8
11	11	7	9	16
12	12	7	10	15
13	13	5	7;D11;D12	26
14	14	5	3;D12	20
15	15	8	8;D13	34
16	16	3	15	37

Найдем максимальное значение D17=**МАКС(D1:D16)**

Ответ: **37**

Решение (ИНДЕКС, через разбиение на столбцы)

Суть решение очень похожа на решение предыдущим способом. Теперь мы будем находить время выполнения процессов с помощью функции ИНДЕКС, которая будет ссылаться на определенную ячейку в столбце с результатами. На первом этапе так же, как и в прошлом методе решения, удаляем все пробелы. Шапку таблицы оставляем, в этом решении нам удобно иметь лишнюю строку.

После этого разбиваем столбец со списком процессов, от которых зависят процессы, по точке с запятой (Данные – Текст по столбцам), в качестве разделителя указываем точку с запятой.

Теперь в столбцах F:H определим значение, соответствующее минимальному времени завершения процесса в столбцах C:E для процесса в текущей строке. Так как мы будем определять время завершения в столбце I (строки с 1 по 16), то указываем такую формулу F2=**ИНДЕКС($I$1:$I$16;C2+1)**. Адреса указываем абсолютные для удобного копирования формулы во все ячейки диапазона F2:H17. Единица добавляется к индексу, так как все номера процессов смещены на одну строку (1 на 2 строке, 2 на 3 и т.д.). Плюс необходимо обработать ситуацию с 0 процессом, в таком варианте он будет «находиться» на первой строке. Пустая ячейка будет интерпретироваться в математической формуле как 0 (в *MS Excel* и *LibreOffice Calc*).

После чего указываем формулу для подсчета минимального времени выполнения процесса I2=**МАКС(F2:H2)+B2**. Желтым и зеленым выделены диапазоны, заполняемые на определенном этапе (желтые – процессы от которых зависит текущий, зеленый – время их завершения).

	A	B	C	D	E	F	G	H	I	J
1	ID процес	Врем	ID процессов А							
2	1	5	0			0	0	0	5	37
3	2	2	0			0	0	0	2	
4	3	3	1	2		5	2	0	8	
5	4	9	3			8	0	0	17	
6	5	7	3			8	0	0	15	
7	6	4	5			15	0	0	19	
8	7	2	4	6		17	19	0	21	
9	8	4	7			21	0	0	25	
10	9	9	0			0	0	0	9	
11	10	8	0			0	0	0	8	
12	11	7	9			9	0	0	16	
13	12	7	10			8	0	0	15	
14	13	5	7	11	12	21	16	15	26	
15	14	5	3	12		8	15	0	20	
16	15	8	8	13		25	26	0	34	
17	16	3	15			34	0	0	37	

С помощью формулы J2=**МАКС(I2:I16)** находим время выполнения самого длинного процесса.

Ответ: **37**

Решение (программное)

Python

```
# скопируем содержимое таблицы в файл
# и прочитаем его.
# Заменим точки с запятой на пробелы
file = open('22.txt').read().replace(';', ' ')
# преобразуем данные в список строк из чисел
nums = [[int(x) for x in s.split()]
        for s in file.split('\n')]
# создадим словарь вида
# №процесса: (время выполнения, процессы)
pr = {c[0]:(c[1], c[2:]) for c in nums}
# определим функцию, которая
# возвращает время работы процесса
# n - номер процесса
def f(n):
    # если номер равен 0, то время выполнения 0
    # (начало всех процессов)
    if n == 0: return 0
    # иначе максимальное время выполнения
    # для всех предыдущиъ процессов
    # плюс время выполнения процесса n
    return max(f(x)
               for x in pr[n][1]) + pr[n][0]

# находим максимальное значение
print(max(f(x) for x in pr))
```

PascalABC.net

Ответ: **37**

23 Исполнитель Аллегро преобразует число на экране.

У исполнителя есть три команды, которым присвоены номера:

1. Прибавить 1
2. Прибавить 2
3. Умножить на 3

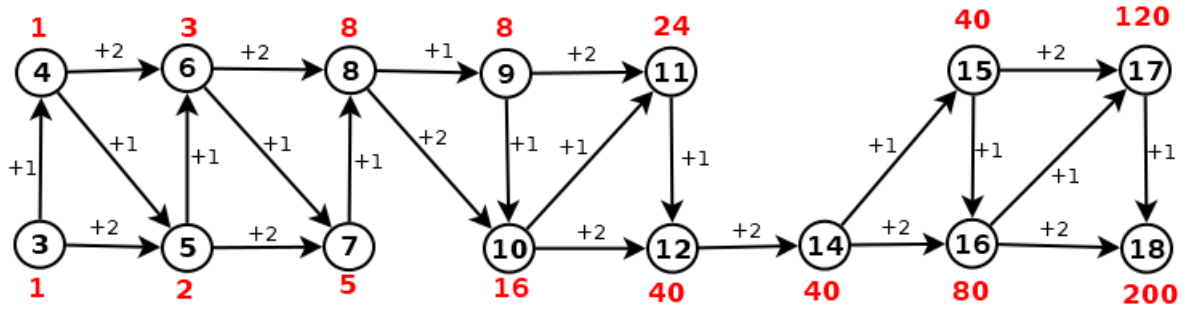
Первая команда увеличивает число на экране на 1, вторая увеличивает число на 2, третья умножает его на 3. Программа для исполнителя Аллегро – это последовательность команд.

Сколько существует программ, для которых при исходном числе 3 результатом является число 18 и при этом траектория вычислений содержит число 8, но не содержит число 13?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы 123 при исходном числе 7 траектория будет состоять из чисел 8, 10, 30.

Решение (ручное, графическое, прямой проход)

Изобразим в виде графа, как можно прийти в определенные числа. И посчитаем количество вычислительных стратегий до каждой вершины из начальной.



Ответ: **200**

Решение (ручное, таблица, прямой проход)

Перечислим все значения в таблице, при этом пометим число 8 и запишем для числа 13 ноль. Количество стратегий нулевой длины – 1, поэтому определим значение для 3 равное единице.

Во второй строке обозначим, из каких значений можно достигнуть текущего за одну команду.

3	**4**	**5**	**6**	**7**	**8**
	3	**3,4**	**4,5**	**5,6**	**6,7**
1	1	1+1=2	1+2=3	2+3=5	3+5=8

8	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
	8	**8,9**	**9,10**	**10,11**		**12**	**14**	**14,15**	**15,16**	**16,17**
1	1	1+1=2	1+2=3	2+3=5	0	5	5	5+5=10	10+5=15	10+15=25

Итого из 3 в 8 имеем 8 маршрутов, из 8 в 18 – 25. Следовательно, общее количество маршрутов из 3 в 18 равно 200.

Ответ: **200**

Решение (электронная таблица, прямой проход)

Выпишем значения от 1 до 18 в строку (можно в столбец, просто скрины будут занимать больше места).

Под 3 поставим значение 1 (начальное). В ячейке D2 запишем такую формулу =B2+C2+если(остат(D1;3)=0;индекс($A2:$R18;D$1/3);0)

C2 и B2, соответственно, количество вычислительных стратегий для значений на 1 и на два больше. Формула =индекс($A2:$R18;D$1/3) находит ячейку, которая соответствует значению в три раза меньшему. И вычисляться эта формула будет только для значений, которые кратны 3.

Копируем формулу в ячейки второй строки до 8 столбца (H), в ячейке H3 записываем формулу =H2 и копируем формулу для вычисления в ячейки диапазона I3:R3. В ячейке M3 (соответствующей значению 13) вставим 0, так как через это значение траектории не должны проходить.

В итоге получим следующую таблицу

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
2	0	0	1	1	2	3	5	8											
3								8	8	16	24	40	0	40	40	80	120	200	
4																			

Ответ: **200**

Решение (программное, динамическое программирование, прямой проход)

Для удобства напишем функцию, которая находит количество вычислительных траекторий из *a* в *b*. Также не забудем прописать условие для пропуска значения 13.

Python

```
def cnt(a, b):
  c = [0]*(b+1)
  c[a] = 1
  for i in range(a, b):
    if i == 13: continue
    for nxt in (i+1, i+2, i*3):
      if nxt <= b:
        c[nxt] += c[i]
  return c[b]
print(cnt(3,8) * cnt(8, 18))
```

PascalABC.net

Ответ: **200**

Решение (программное, рекурсивное, прямой проход)

Также напишем функцию и учтем случай для *a*=13.

Python

```
def cnt(a, b):
  if a > b or a == 13:
    return 0
  if a == b:
    return 1
  return cnt(a+1, b)+cnt(a+2, b)+cnt(a*3, b)
print(cnt(3,8) * cnt(8, 18))
```

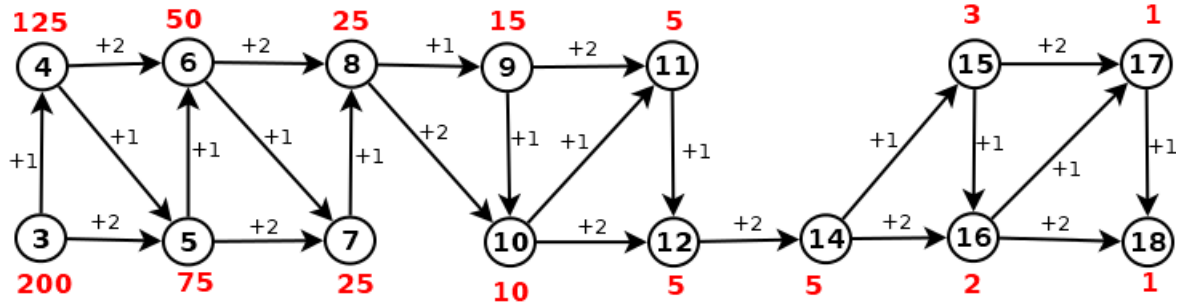
PascalABC.net

```
## // Автор: Платорн Коктышев
function cnt(a,b:integer):integer
  := (a > b)or(a = 13)
     ? 0
     : a = b
       ? 1 : cnt(a+1, b)+cnt(a+2, b)+ cnt(a*3, b);
print(cnt(3,8) * cnt(8, 18));
```

Ответ: **200**

Решение (ручное, графическое, обратный проход)

Изобразим в виде графа, как можно прийти из всех чисел в 18. И посчитаем количество вычислительных стратегий от каждой вершины до конечной.



Ответ: **200**

Решение (ручное, таблица, обратный проход)

Перечислим все значения в таблице, при этом пометим число 8 и запишем для числа 13 ноль. Количество стратегий нулевой длины – 1, поэтому определим значение для 8 и 18 равными единице.

Во второй строке обозначим, в какое числа из заданного диапазона можно попасть из текущего числа.

3	**4**	**5**	**6**	**7**	**8**
4,5	**5,6**	**6,7**	**7,8**	**8**	
5+3=8	2+3=5	1+2=3	1+1=2	1	1

8	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
9,10	**10,11**	**11,12**	**12**	**14**	**X**	**15,16**	**16,17**	**17, 18**	**18**	
10+15=25	5+10=15	5+5=10	5	5	0	2+3=5	1+2=3	1+1=2	1	1

Итого из 8 в 18 имеем 25 маршрутов, из 3 в 8 – 8. Следовательно, общее количество маршрутов из 3 в 18 равно 200.

Ответ: **200**

Решение (электронная таблица, обратный проход)

Выпишем значения от 1 до 18. Важно, чтобы номера столбцов совпадали с числами в строке (A-1, B-2 и т.д.)

Под 18 поставим значение 1 (начальное). В ячейке Q2 запишем такую формулу =R2+S2+ИНДЕКС($A2:$RR18; Q$1*3) после чего скопируем формулу в диапазон H2:Q2. Важно, чтобы диапазон было больше 18 значений, чтобы все значения для команды *3 имели соответствие определенному столбцу. В ячейку, соответствующую числу 13, записываем 0.

В ячейке H3 записываем формулу =H2 и копируем формулу для вычисления в ячейки диапазона C3:G3.

В итоге получим следующую таблицу

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
2								25	15	10	5	5	0	5	3	2	1	1	
3			200	125	75	50	25	25											

Ответ: **200**

Решение (программное, динамическое программирование, обратный проход с вычислением по уже вычисленным)

Для удобства напишем функцию, которая находит количество вычислительных траекторий из *a* в *b*. Также не забудем прописать условие для пропуска значения 13.

Python	**PascalABC.net**

`def cnt(a, b):`
`  c = [0]*(b+1)`
`  c[b] = 1`
`  for i in range(b-1, a-1, -1):`
`  if i == 13: continue`
`  for nxt in (i+1, i+2, i*3):`
`  if nxt <= b:`
`  c[i] += c[nxt]`
`  return c[a]`
`print(cnt(3,8) * cnt(8, 18))`	

Ответ: **200**

Решение (программное, динамическое программирование, обратный проход с вычислением из вычисленных значений)

Python	**PascalABC.net**

`def cnt(a, b):`
`  c = [0]*(b+1)`
`  c[b] = 1`
`  for i in range(b, a, -1):`
`  if i == 13: continue`
`  for nxt in [i-1,i-2]\`
`  +([i//3] if i%3 == 0 else []):`
`  if nxt >= a:`
`  c[nxt] += c[i]`
`  return c[a]`
`print(cnt(3,8) * cnt(8, 18))`	

Ответ: **200**

Решение (программное, рекурсивное, обратный проход)

Также напишем функцию и учтем случай для *a*=13.

Python	**PascalABC.net**

`def cnt(a, b):`
`  if a > b or b == 13:`
`  return 0`
`  if a == b:`
`  return 1`
`  return cnt(a, b-1)+cnt(a, b-2) \`
`  +(cnt(a, b//3) if b % 3 == 0 else 0)`
`print(cnt(3,8) * cnt(8, 18))`
	`## // Автор: Платон Коктышев`
	`function cnt(a,b:integer):integer :=`
	`  (a > b) or (b = 13)`
	`  ? 0`
	`  : a = b`
	`  ? 1`
	`  :cnt(a, b-1)+cnt(a, b-2)+(cnt(a, b div 3));`
	`print(cnt(3,8) * cnt(8, 18));`

Ответ: **200**

24 ***Задание выполняется с использованием прилагаемых файлов.***

Текстовый файл состоит не более чем из 1 200 000 символов английского алфавита

Определите максимальное количество идущих подряд символов, среди которых символы *Q*, *R*, *S* в различных комбинациях (с учётом повторений) не стоят рядом.

Для выполнения этого задания следует написать программу.

Решение (программное, динамическое программирование)

Python

```
s = open('24.txt').readline()
# длина подходящей последовательности для s[0]
# она же – текущее максимальное значение длины
c = mx = 1
for i in range(1, len(s)):
  # если хотя бы один символ s[i] и s[i-1]
  # не является символами Q, R, S
  if s[i-1] not in 'QRS' or s[i] not in 'QRS':
    # увеличиваем текущую длину строки
    c += 1
    # переопределяем максимальное значение
    mx = max(mx, c)
  else:
    # если оба символа входят в Q, R, S
    # начинаем последовательность с
    # текущего символа
    c = 1
print(mx)
```

PascalABC.net

Ответ: **544**

Решение (программное, замена запрещенных комбинаций)

Идея решения заключается в том, чтобы разбить строку на слова, разделенные пробелами, в которых нет запрещенных пар символов.

Python

```
s = open('24.txt').readline()
# для всех запрещенных комбинаций a+b
for a in 'QRS':
  for b in 'QRS':
    while a+b in s:
      # вставляем между элементами пары пробел
      # теперь имеем строку из слов
      # в которых нет запрещенных пар символов
      s = s.replace(a+b, a+' '+b)
print(max(map(len, s.split())))
```

PascalABC.net

```
## // Автор: Максим Крючков
var t:=ReadAllText('24.txt');
t:=Regex.Replace(t, '[QRS][QRS]', 'Q Q');
t.Split.select(s -> s.length).Max.print;
```

Ответ: **544**

Решение (программное, упрощение предыдущего метода)

Так как нет необходимости сохранить строку в исходном виде, можно заменить все неподходящие символы на неиспользуемый символ, например, * и потом разделить все подряд идущие пары ** пробелом.

Python

```
s = open('24.txt').readline()
for c in 'QRS':
    s = s.replace(c, '*')
while '**' in s:
    s = s.replace('**', '* *')
print(max(map(len, s.split())))
```

PascalABC.net

Ответ: **544**

Решение (программное, перебор срезов, долгое)

Идея решения: перебрать все возможные длины срезов до длины, для которой не найдется ни одной подстроки в строке, в которой не было бы запрещенной комбинации.

Python	PascalABC.net

```
s = open('24.txt').readline()
# множество запрещенных пар
qrs = set(a+b for a in 'QRS' for b in 'QRS')
# перебираем все допустимые длины срезов
for c in range(1, len(s)):
  # перебираем все допустимые индексы начала
  # для срезов длиной c+1
  for i in range(len(s) - c):
    # если в срезе нет запрещенных комбинаций
    if all(x not in s[i: i+c+1] for x in qrs):
      # прерываем внутренний цикл
      break
  # если цикл ни разу не прервался
  else:
    # прерываем внешний, так как не нашли
    # ни одну строку без запрещенных пар
    break
# выводим длину (c – значение на 1 меньше, чем
# перебранные длины, поэтому значение, при
# котором алгоритм вышел из цикла, совпадает с
# искомым значением длины)
print(c)
```

Примечание: для длины 544 это достаточно «быстрый» поиск. Так как поиск подстроки в строке имеет сложность O(n), то чем больше длина подходящей строки, тем медленнее будет происходить поиск по такому алгоритму.

Ответ: **544**

Решение (программное, перебор срезов, бинарный поиск, чуть быстрее в общем случае)

Идея «ускорения»: перебирать не все длины срезов, а сокращать или увеличивать их длину, уменьшая диапазон перебора вдвое.

Python	PascalABC.net

```
s = open('24.txt').readline()
# множество запрещенных пар
qrs = set(a+b for a in 'QRS' for b in 'QRS')

st = 1
fn = len(s)
while st != fn:
  hlf = (st+fn) // 2
  for i in range(len(s) - hlf):
    if all(x not in s[i: i+hlf+1]
           for x in qrs):
      # если для длины+1 нашелся срез,
      # то искомая длина не меньше
      st = hlf + 1
      break
  else:
    # если подходящих строк не нашлось
    # то искомая длина меньше hlf+1
    fn = hlf
print(st)
```

Примечание: в конкретном условии из досрочного экзамена 2023 такая оптимизация скорее замедляет поиск. Однако, если длина искомой строки будет в разы больше, то разница будет заметна (для длины в 1_000_000 такой алгоритм находит искомую длину не более, чем за 20 итераций).

Ответ: **544**

Ответ: _____.

25 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 10^8, найдите все числа, соответствующие маске 12??36*1, делящиеся на 273 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 273.

Решение (через перебор вариантов и составление числа)
Заметим, что при ограничении до 10^8 на месте звездочки может стоять не больше одного разряда. Поэтому достаточно перебрать два варианта – пустая строка на месте звездочки и 1 символ на месте звездочки.

Python

```
for a in range(10):
  for b in range(10):
    x = 12*10**5 + a*10**4 + b*10**3 + 361
    # можно заменить на
    # x = int(f'12{a}{b}361')
    if x % 273 == 0:
      print(x, x // 273)

for a in range(10):
  for b in range(10):
    for c in range(10):
      x = 12*10**6 + a*10**5 + b*10**4\
          + 36*100 + c*10 + 1
      # можно заменить на
      # x = int(f'12{a}{b}36{c}1')
      if x % 273 == 0:
        print(x, x // 273)
```

PascalABC.net

Примечание: на мой личный взгляд это пример каноничного решения задач на перебор.

Решение (анализ строки без регулярных выражений)
// Автор: Алёна Смирнова
Перебираемый диапазон
А) диапазон для поиска нужных значений:
1) минимальное число, подходящее под маску: 1200361
2) максимальное число, подходящее под маску: 12993691
=> искать ответ будем в диапазоне [1200361; 12993691]
Б) минимальное число, входящее в диапазон, кратное 273:
1200361 / 273 = 4396.9267... => 4397*273 - искомое значение
Проверка по маске
Так как мы имеем одну звездочку, то будем проверять окончание после нее и часть до нее:
- первые две цифры 12,
- пятая и шестая 36,
- последняя 1.

Python

```
# перебираем числа из найденного диапазона
for x in range(4397*273, 12993691+1, 273):
    n = str(x)
    # проверяем, подходит ли число под маску:
    if n[:2],n[4:6],n[-1] == '12', '36', '1':
        print(x, x//273)
```

PascalABC.net

Решение (программное, через библиотеку для работы с масками)

Если перебирать все числа, то можно начать от нуля с шагом 273. Именно так можно найти все числа, кратные 273. Затем каждое преобразовать в строку и проверить на соответствие маске из задания.

Python

```
from fnmatch import fnmatch

for x in range(0, 10**8, 273):
  if fnmatch(str(x), '12??36*1'):
    print(x, x // 273)
```

PascalABC.net

```
### // Автор: Инна Свистун
for var n:=1200361 to 12993601 do
begin
 if (n mod 10=1)and (n mod 273=0)then
  if(n.ToString[:3]='12')and (n.ToString[5:7]='36')
  then  prln(n, n div 273);
end
```

Примечание: такой способ хорош, когда требуется проверить по маске сопоставимое с 10^6-10^8 строк. Если перед вами стоит необходимость обработать большее количество строк, стоит задуматься над методами оптимизации поиска.

Решение (программное, через регулярные выражения)

Рассуждение аналогично предыдущему решению. Вместо библиотеки для работы с файловыми масками используется библиотека для работы с регулярными выражениями.

Python

```
from re import match

for x in range(0, 10**8, 273):
  if match(r'^12..36.*1$', str(x)):
    print(x, x // 273))
```

PascalABC.net

```
## // Автор: Максим Крючков
Range(273, 100000000, 273)
 .where(i -> i.ToString.IsMatch('^12..36.*1$'))
 .Select(i -> (i, i/273)).PrintLines;
```

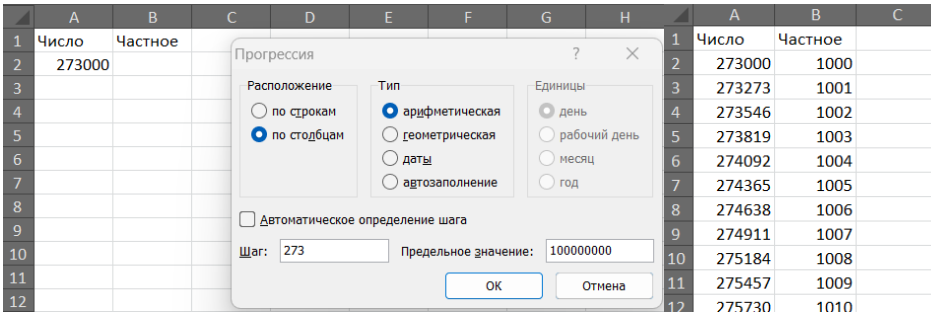
Ответ:

1271361	**4657**
16233621	**46277**
12663651	**46387**
12693681	**46497**

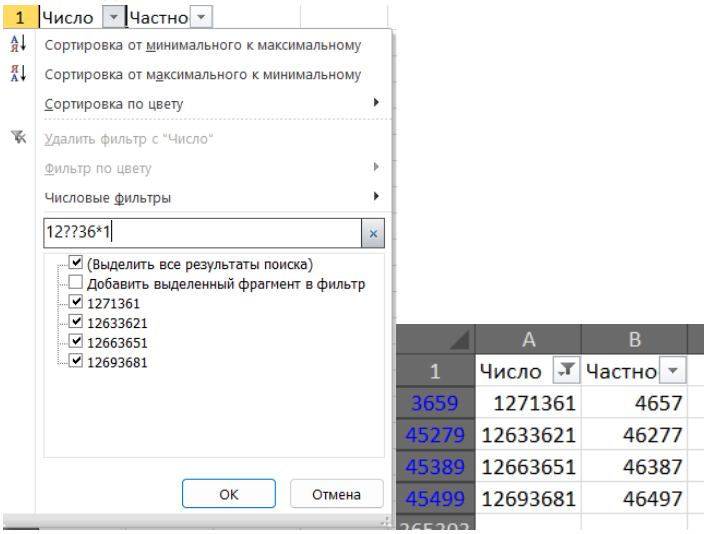
Решение (электронные таблицы)

Автор: Эльвира Малышева

В столбце А вставим арифметическую прогрессию для чисел, кратных 273. Так как мы ищем числа от 7 знаков (минимальная длина по маске), то имеет смысл начать прогрессию со значения, которое по длине расположено как можно ближе. Самый простой вариант – 273000. И заполним второй столбец частными от деления найденных чисел на 273



Теперь применим фильтр в соответствии с заданием



26 ***Задание выполняется с использованием прилагаемых файлов.***

В аэропорту есть камера хранения из K ячеек, которые пронумерованы с 1. Принимаемый багаж кладется в свободную ячейку с минимальным номером. Известно время, когда пассажиры сдают и забирают багаж (в минутах с начала суток). Ячейка доступна для багажа, начиная со следующей минуты, после окончания срока хранения. Если свободных ячеек не находится, то багаж не принимается в камеру хранения.
Найдите количество багажа, которое будет сдано в камеры за 24 часа и номер ячейки, в которую сдаст багаж последний пассажир.
Входные данные
В первой строке входного файла находится число K – количество ячеек в камере хранения, во второй строке файла число N – количество пассажиров, сдающих багаж (натуральное число, не превышающее 1000). Каждая из следующих N строк содержит два натуральных числа, не превышающих 1440: время сдачи багажа и время выдачи багажа.
Выходные данные
Программа должна вывести два числа: количество сданных в камеру хранения багажа и номер ячейки, в которую примут багаж у последнего пассажира, который сможет сдать багаж.

Типовой пример организации данных:
2
5
30 60
40 60
50 1110
61 1010
1100 1440

Для указанного примера багаж смогут сдать первый, второй, четвёртый и пятый пассажир. Последний пассажир сдаст свой багаж в ячейку один, так как к этому моменту первая и вторая ячейка будут свободны.

Решение (без сортировки):
Описание идеи. Опишем хранилище в виде списка (массива), где изначально будут сохранены значения -1 (не занято).
Дальше пройдем по таймлайну и для каждой минуты (от 0-й до 1439-й) будем проверять:
- есть ли багаж, который нужно сдать в эту минуту (в задаче нет описания разрешения конфликтов, поэтому считаем, что в одну минуту не может приниматься несколько багажей),
- есть ли багажи, которые выдаются в эту минуту.
Если есть багаж, который может быть сдан в текущую минуту, проверяем, есть ли для него свободная ячейка. В качестве ячейки приема выбираем первую по возрастанию номера. Увеличиваем счетчик сданных багажей и запоминаем номер ячейки, куда сдали багаж. В качестве значения в хранилище определим время выдачи соответствующего багажа.
Если есть багаж (один или несколько), который в эту минуту должен выдать, то сохраняем в качестве значения в хранилище -1 для такого багажа.
Проверка наличия свободной ячейки и опустошение ячеек происходит в таком порядке, чтобы при выдаче багажа положить в освободившуюся ячейку багаж можно было только на следующей итерации.

Python

```
with open('26.txt') as f:
    k = int(f.readline())
    n = int(f.readline())
    nums = [[int(x)
             for x in f.readline().split()]
            for _ in range(n)]
storage = [-1]*(k+1)  # хранилище
last = cnt = 0
for time in range(1440): # перебираем минуты
  for i in range(n):
    if nums[i][0] == time: # если сдают багаж
      for cell in range(1,k+1):
        if storage[cell] == -1:# пустая ячейка
          storage[cell],last = nums[i][1],cell
          cnt += 1
          break
  for cell in range(k):
    if storage[cell] == time: # багаж забирают
      storage[cell] = -1 # освобождаем ячейку
print(cnt, last)
```

PascalABC.net

Ответ: **586 3**

Решение (без сортировки, с использованием словаря):
Для наиболее удобного поиска багажа заведем словарь вида «время сдачи»: «времена выдачи». Так как в задаче не указано, что нет багажей, которые сдаются одновременно, необходимо предусмотреть хранение нескольких багажей с одинаковым временем сдачи. При таком изменении мы можем определять есть ли багажи с проверяемым временем сдачи за O(1) с помощью команды *time in bags*.

Python

```
from collections import defaultdict
with open('26.txt') as f:
    k = int(f.readline())
    n = int(f.readline())
    bags = defaultdict(list)
    for i in range(n):
      a, b = map(int, f.readline().split())
      bags[a].append(b)
storage = [-1]*(k+1)  # хранилище
last = cnt = 0
for time in range(1440): # перебираем минуты
  if time in bags: # есть багаж в очереди
    for end in bags[time]:
      for cell in range(1,k+1):
        if storage[cell] == -1:# пустая ячейка
          storage[cell], last = end, cell
          cnt += 1
          break
  for cell in range(k):
    if storage[cell] == time: # багаж забирают
      storage[cell] = -1 # освобождаем ячейку
print(cnt, last)
```

PascalABC.net

Ответ: **586 3**

Решение (сортировка + заполнение таймлайнов)
Создадим K таймлайнов по 1440 минут. И отсортируем багажи по времени сдачи в хранилище. Если текущая минута свободна, будем записывать таймлайн первой свободной ячейки набор единиц длиной равной времени хранения багажа, увеличенной на 1. Для каждого сохраненного багажа будем запоминать номер ячейки и увеличивать счетчик сданных багажей.
Так как мы упорядочили багаж по времени сдачи, то достаточно проверять на таймлайне свободна ли минута сдачи очередного багажа.

Python

```
with open('26.txt') as f:
    k = int(f.readline())
    n = int(f.readline())
    nums = [[int(x)
             for x in f.readline().split()]
            for _ in range(n)]
nums.sort()
times = [[0]*1441 for _ in range(k+1)]
c = last = 0
for st, fn in nums:
  for t in range(1, k+1):
    if times[t][st] == 0:
      times[t][st:fn+1] = [1]*(fn+1 - st)
      c += 1
      last = t
      break
print(c, last)
```

PascalABC.net

Примечание: пример таймлайна для трех ячеек и сданных багажей (3 6), (4 10), (5 8), (9 12)

		Текущая минута											
		1	2	3	4	5	6	7	8	9	10	11	12
Номер ячейки	1	-1	-1	1	1	1	1	-1	-1	1	1	1	1
	2	-1	-1	-1	1	1	1	1	1	1	1	-1	-1
	3	-1	-1	-1	-1	1	1	1	1	-1	-1	-1	-1

Ответ: **586 3**

Решение (сортировка + сохранение только времени выдачи багажа)
На примере предыдущего решения можно заметить, что достаточно знать время освобождения ячейки, чтобы определить можно ли в нее положить следующий багаж. Поэтому изменим таймлайн на одномерный список, в котором будем хранить либо время освобождения ячейки, либо 0, если ячейка свободна. Строгое неравенство, так как между выдачей и сдачей багажа должна пройти 1 минута.

Python

```
with open('26.txt') as f:
    k = int(f.readline())
    n = int(f.readline())
    nums = [[int(x)
             for x in f.readline().split()]
            for _ in range(n)]
nums.sort()

times = [0]*(k+1)
c = last = 0
for st, fn in nums:
  for t in range(1, k+1):
    # если ячейка уже свободна
    if times[t] < st:
      # запоминаем, когда ячейка освобождается
      times[t] = fn
      c += 1
      last = t
      break
print(c, last)
```

PascalABC.net

```
## // Автор: Алексей Богданов
var lines := ReadAllLines('26.txt');
var (k,n) := lines[:2].Select(s->s.ToInteger);
var a := lines.Skip(2).Select(s -> s.ToIntegers)
          .Select(\(x,y)->(x,y)).Order.ToArray;
var z := |-1|*k;
var (last,t) := (0,0);
foreach var (x,y) in a do
  foreach var f in z index i do
    if f<x then begin
      z[i] := y;
      last := i;
      t += 1;
      break
    end;
print(t,last+1);
```

Ответ: **586 3**

27 *Задание выполняется с использованием прилагаемых файлов.*

Метеорологическая станция ведёт наблюдение за количеством выпавших осадков. Показания записываются каждую минуту в течении N минут. Определяется пара измерений, между которыми прошло не менее K минут. Найдите максимальную сумму показаний среди таких пар.

Входные данные
Даны два входных файла (A и B), каждый из которых в первой строке содержит число N - количество измерений, во второй строке K - минимальное количество минут между искомыми измерениями. В каждой из следующих N строк находится число: количество выпавших осадков.

В ответе укажите два числа: сначала значение искомой величины для файла A, затем - для файла B.

Типовой пример организации данных во входном файле
5
3
10
15
100
1
30

При таких исходных данных ответом будет 45

Предупреждение: для обработки файла B не следует использовать переборный алгоритм, вычисляющий сумму для всех возможных вариантов, поскольку написанная по такому алгоритму программа будет выполняться слишком долго.

Решение (27А, полный перебор)

Python

```
with open('27A.txt') as f:
  n = int(f.readline())
  k = int(f.readline())
  nums = [int(f.readline()) for _ in range(n)]
# количество не может быть отрицательным
mx = 0
# перебираем все индексы в парах
for i in range(n):
  for j in range(i+k, n):
    # переопеделяем максимум если надо
    mx = max(mx, nums[i] + nums[j])
# выводим максимум
print(mx)
```

PascalABC.net

```
## // Автор: Инна Свистун
var F:=openRead('27A.txt');
var n:= F.ReadInteger; var k:=F.ReadInteger;
var a:= (1..n).Sel(x->F.ReadInteger).ToA;
var sMax :=0;
for var i:=0 to a.H-k-1 do
  for var j:=i+k to a.H do
    if (a[i]+a[j]>sMax) then sMax:=a[i]+a[j];
Pr(sMax);
```

Ответ: **174902**

Решение (27Б, неоптимальное по памяти со списком максимумов)

// Идея: Алёна Смирнова

Идея заключается в следующем:

Максимальная сумма пары в списке *nums* с числом, стоящим на *i+k* позиции, будет получена в результате сложения с максимальным значением на срезе *nums[:i+1]*. Или *max(nums[:i+1]) + nums[i+k]*

Поэтому мы можем создать еще один список mx, где значение *mx[i]* будет хранить максимальное значение на срезе *nums[:i+1]*

Python

```
n, k, *nums = map(int, open('27B.txt'))
# mx - список для хранения
# значения max(nums[:i+1])
mx = [0]*n
mx[0] = nums[0]
for i in range(1, n):
    mx[i] = max(mx[i-1], nums[i])
print(max(mx[i]+nums[i+k]
          for i in range(n-k)))
```

PascalABC.net

Ответ: **3094684**

Решение (27Б, неоптимальное по памяти)

При решении будем руководствоваться рассуждением:

- перебираем числа, начиная с K+1
- для каждого перебираемого числа нам нужно только одно число из начала последовательности – максимум на расстоянии ≥K.

Python

```
with open('27B.txt') as f:
  n = int(f.readline())
  k = int(f.readline())
  nums = [int(f.readline()) for _ in range(n)]
# количество не может быть отрицательным
mx_left = mx = 0
for i in range(k, n):
  # переопределяем максимум
  # в начале последовательности
  mx_left = max(mx_left, nums[i-k])
  # переопеделяем максимум если надо
  mx = max(mx, nums[i] + mx_left)
# выводим максимум
print(mx)
```

PascalABC.net

```
### // Автор: Инна Свистун
var F:=openRead('27B.txt');
var n:= F.ReadInteger; var k:=F.ReadInteger;
var a:= (1..n).Sel(x->F.ReadInteger).ToA;
var (sMax,MaxLeft):=(0,0);
foreach var i in (0..a.H) do
  begin
    if (MaxLeft<>0) and (a[i]+MaxLeft>sMax)
    then sMax:=a[i]+MaxLeft;
    if (i+1-k>=0)and (a[i+1-k]>MaxLeft)
    then MaxLeft:=a[i+1-k];
  end;
Pr(sMax);
```

Ответ: **3094684**

Решение (27Б, сохраняем только K значений)

Для удобного хранения применим очередь со сдвигом начала очереди.

Python	PascalABC.net

```
with open('27B.txt') as f:
    n = int(f.readline())
    k = int(f.readline())
    nums = [int(f.readline())
            for _ in range(k)]
    # количество не может быть отрицательным
    mx_left = mx = 0
    for i in range(k, n):
      x = int(f.readline())
      # переопределяем максимум
      # в начале последовательности
      mx_left = max(mx_left, nums[i%k])
      # переопеделяем максимум если надо
      mx = max(mx, x + mx_left)
      # на место числа на K левее записываем x
      nums[i%k] = x
    # выводим максимум
    print(mx)
```

Ответ: **3094684**

№ задания	Ответ
1	**7**
2	**xwzy**
3	**455**
4	**100**
5	**22**
6	**77**
7	**1080**
8	**251**
9	**853**
10	**271**
11	**4992**
12	**16**
13	**66**

№ задания	Ответ
14	**116070624**
15	**36**
16	**1338**
17	**2** **33120**
18	**1102** **1029**
19	**19**
20	**15** **18**
21	**14**
22	**37**
23	**200**
24	**544**
25	**1271361** **4657** **16233621** **46277** **12663651** **46387** **12693681** **46497**
26	**586** **3**
27	**174902** **3094684**

Система оценивания экзаменационной работы по информатике и ИКТ

За правильный ответ на задания 1–25 ставится 1 балл; за неверный ответ или его отсутствие – 0 баллов.

За верный ответ на задание 26 ставится 2 балла; если значения в ответе перепутаны местами ИЛИ в ответе присутствует только одно верное значение (второе неверно или отсутствует) – ставится 1 балл. В остальных случаях – 0 баллов.

За верный ответ на задание 27 ставится 2 балла; если значения в ответе перепутаны местами ИЛИ в ответе присутствует только одно верное значение (второе неверно или отсутствует) – ставится 1 балл. В остальных случаях – 0 баллов.

Файлы к варианту: https://drive.google.com/drive/folders/1YtoxgG3BXGl3NdJ0q6wJc_16TXhViZ-n

Ссылка на тест в эмуляторе: https://kompege.ru/variant?kim=25023734

Информация об авторе

Составили	Алексей Кабанов ВК https://vk.com/ege_info_open Youtube www.youtube.com/user/axelofan2010
	Евгений Джобс ВК https://vk.com/inform_web Youtube https://www.youtube.com/channel/UCu50NY1uYmfuAWtNqPpHyDg
Автор эмулятора	Алексей Кабанов ВК vk.com/cabanovalexey Youtube www.youtube.com/user/axelofan2010