

In[1]:= **Составить список из заглавных букв английского алфавита (в кодировке Unicode), посчитать среднее значение кодов.**

Syntax:

"Составить список из заглавных букв английского алфавита (в кодировке Unicode), посчитать среднее значение кодов." is incomplete; more input is needed.

```
In[1]:= AlphabetChar = CharacterRange["A", "Z"]
CharCode = ToCharacterCode[StringJoin[AlphabetChar]]
CharSum = Sum[CharCode[[i]], {i, 1, Length[CharCode]}]
CharAverage = N[CharSum/Length[CharCode]]
```

Out[1]= {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}

Out[2]= {65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75,
76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90}

Out[3]= 2015

Out[4]= 77.5

In[5]:= **Дано число 358 138 [number],
найти его обратное число [res1] в конечном поле GF (6 502 301) [GF (div)].**

Syntax: "Дано число 358138[number], найти его обратное число[res1] в конечном поле GF (6502301)[GF (div)]." is incomplete; more input is needed.

```
In[8]:= number = 358 138
div = 6 502 301
res1 = PowerMod[number, -1, div]
```

Out[8]= 358 138

Out[9]= 6 502 301

Out[10]= 4 060 720

In[11]:= **Дано число 7706 [np], найти два ближайших к нему
простых числа (b1, b2). Посчитать их сумму по модулю 131 [m].**

Syntax:

"Дано число 7706[np], найти два ближайших к нему простых числа (b1, b2).Посчитать их сумму по модулю 131[m]." is incomplete; more input is needed.

```
In[11]:= np = 7706
         m = 131
         b1 = NextPrime[np, -1]
         b2 = NextPrime[np, +1]
         SumPrime = Mod[(b1 + b2), m]
```

```
Out[11]= 7706
```

```
Out[12]= 131
```

```
Out[13]= 7703
```

```
Out[14]= 7717
```

```
Out[15]= 93
```

```
In[16]:= Найти количество разрядов в двоичной записи шестнадцатиричного числа 33 A855.
```

Syntax: "A855." is incomplete; more input is needed.

```
In[18]:= Des = 16^^33A855
         LenDv = IntegerLength[Des, 2]
```

```
Out[18]= 3 385 429
```

```
Out[19]= 22
```

```
In[20]:= Найти количество простых чисел в диапазоне от 17 [R1] до 29 [R2].
```

Syntax: "29[R2]." is incomplete; more input is needed.

```
In[20]:= R1 = 17
         R2 = 29
         ListR = Table[s, {s, R1, R2}]
         If[PrimeQ[R1], BeforeR1 = PrimePi[R1] - 1, BeforeR1 = PrimePi[R1]]
         BeforeR2 = PrimePi[R2]
         CountPrime = BeforeR2 - BeforeR1
```

```
Out[20]= 17
```

```
Out[21]= 29
```

```
Out[22]= {17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29}
```

```
Out[23]= 6
```

```
Out[24]= 10
```

```
Out[25]= 4
```

```
In[26]:= Получить множество целых чисел от [min1] до [max1] и множество простых чисел от [min2] до [max2]. Найти произведение элементов пересечения этих множеств.
```

Syntax: "множеств." is incomplete; more input is needed.

```

In[26]:= min1 = 65
         max1 = 105
         List1 = Range[min1, max1]

Out[26]= 65

Out[27]= 105

Out[28]= {65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86,
         87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105}

In[34]:= min2 = 80
         max2 = 145
         If[PrimeQ[min2], t1 = PrimePi[min2], t1 = PrimePi[min2]+1]
         t2 = PrimePi[max2]
         List2 = Table[Prime[x], {x, t1, t2}]

Out[34]= 80

Out[35]= 145

Out[36]= 23

Out[37]= 34

Out[38]= {83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139}

In[39]:= ListM = Intersection[List1, List2]
         LenListM = Length[ListM]

Out[39]= {83, 89, 97, 101, 103}

Out[40]= 5

In[41]:= mult = 1; For[j = 1, j ≤ LenListM, j++, mult *= ListM[[j]]]
         Print[mult]

7 454 155 217

```

Составить список из нечетных чисел в диапазоне от `nmin` до `nmax`.

Сделать квадратную матрицу. Поменять местами строки `str1` и `str2`.

Циклично сдвинуть строку `str3` вправо (влево) на `len1`.

Поменять местами столбцы `col1` и `col2`.

Циклично сдвинуть столбец `col3` вверх (вниз) на `len2`.

Посчитать произведение элементов главной диагонали.

Syntax: "диагонали." is incomplete; more input is needed.

```

In[43]:= nmin = 1
         nmax = 32
         ListB = Table[2 * b - 1, {b, Ceiling[nmin/2], Floor[nmax/2]}]

Out[43]= 1

Out[44]= 32

Out[45]= {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31}

In[46]:= SqrtB = Sqrt[Length[ListB]]
         MatrixB = Partition[ListB, SqrtB]
         Dimensions[MatrixB, SqrtB]
         MatrixB // MatrixForm

Out[46]= 4

Out[47]= {{1, 3, 5, 7}, {9, 11, 13, 15}, {17, 19, 21, 23}, {25, 27, 29, 31}}

Out[48]= {4, 4}

Out[49]//MatrixForm=

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \\ 25 & 27 & 29 & 31 \end{pmatrix}$$


In[50]:= str1 = 2
         str2 = 4
         MatrixC = MatrixB
         MatrixC[[str1]] = MatrixB[[str2]]
         MatrixC[[str2]] = MatrixB[[str1]]
         MatrixB = MatrixC
         MatrixB // MatrixForm

Out[50]= 2

Out[51]= 4

Out[52]= {{1, 3, 5, 7}, {9, 11, 13, 15}, {17, 19, 21, 23}, {25, 27, 29, 31}}

Out[53]= {25, 27, 29, 31}

Out[54]= {9, 11, 13, 15}

Out[55]= {{1, 3, 5, 7}, {25, 27, 29, 31}, {17, 19, 21, 23}, {9, 11, 13, 15}}

Out[56]//MatrixForm=

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 25 & 27 & 29 & 31 \\ 17 & 19 & 21 & 23 \\ 9 & 11 & 13 & 15 \end{pmatrix}$$


```

```
In[57]:= str3 = 3
len1 = 1
MatrixB[[str3]] = RotateRight[MatrixB[[str3]], len1]
MatrixB // MatrixForm
```

```
Out[57]= 3
```

```
Out[58]= 1
```

```
Out[59]= {23, 17, 19, 21}
```

```
Out[60]//MatrixForm=
```

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 25 & 27 & 29 & 31 \\ 23 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 \end{pmatrix}$$

```
In[61]:= col1 = 1
col2 = 3
MatrixD1 = Transpose[MatrixB]
MatrixD2 = MatrixD1
MatrixD2[[col1]] = MatrixD1[[col2]]
MatrixD2[[col2]] = MatrixD1[[col1]]
MatrixB = Transpose[MatrixD2]
MatrixB // MatrixForm
```

```
Out[61]= 1
```

```
Out[62]= 3
```

```
Out[63]= {{1, 25, 23, 9}, {3, 27, 17, 11}, {5, 29, 19, 13}, {7, 31, 21, 15}}
```

```
Out[64]= {{1, 25, 23, 9}, {3, 27, 17, 11}, {5, 29, 19, 13}, {7, 31, 21, 15}}
```

```
Out[65]= {5, 29, 19, 13}
```

```
Out[66]= {1, 25, 23, 9}
```

```
Out[67]= {{5, 3, 1, 7}, {29, 27, 25, 31}, {19, 17, 23, 21}, {13, 11, 9, 15}}
```

```
Out[68]//MatrixForm=
```

$$\begin{pmatrix} 5 & 3 & 1 & 7 \\ 29 & 27 & 25 & 31 \\ 19 & 17 & 23 & 21 \\ 13 & 11 & 9 & 15 \end{pmatrix}$$

```

In[69]:= col3 = 2
len2 = 3
MatrixF = Transpose[MatrixB]
MatrixF[[col3]] = RotateLeft[MatrixF[[col3]], len2]
MatrixB = Transpose[MatrixF]
MatrixB // MatrixForm

Out[69]= 2

Out[70]= 3

Out[71]= {{5, 29, 19, 13}, {3, 27, 17, 11}, {1, 25, 23, 9}, {7, 31, 21, 15}}

Out[72]= {11, 3, 27, 17}

Out[73]= {{5, 11, 1, 7}, {29, 3, 25, 31}, {19, 27, 23, 21}, {13, 17, 9, 15}}

Out[74]//MatrixForm=

$$\begin{pmatrix} 5 & 11 & 1 & 7 \\ 29 & 3 & 25 & 31 \\ 19 & 27 & 23 & 21 \\ 13 & 17 & 9 & 15 \end{pmatrix}$$


```

```

In[84]:= ListD = Table[1, {g, 1, SqrtB}]
MultD = 1; For[u = 1, u ≤ SqrtB, u++, ListD[[u]] = MatrixB[[u, u]];
MultD *= ListD[[u]]]; ListD
Print[MultD]

Out[84]= {1, 1, 1, 1}

Out[85]= {5, 3, 23, 15}

5175

```

In[87]:= **Установить генератор случайных чисел в начальное состояние с параметром (par) $2^{15} \pmod{311}$. Получить список из 10 случайных простых чисел в диапазоне от 46 до 77. Найти произведение двух простых чисел, встречающихся в списке с максимальной (maxfr) и минимальной (minfr) частотами. В случае наличия чисел с одинаковыми частотами, выбирать первые в списке.**

Syntax:

"Установить генератор случайных чисел в начальное состояние с параметром (par) $2^{15} \pmod{311}$. Получить список из 10 <<9>> простых чисел в диапазоне от 46 до 77. Найти произведение двух простых чисел, <<1>>, <<1>>" is incomplete; more input is needed.

```

In[89]:= par = PowerMod[2, 15, 311]
SeedRandom[par]

```

```

Out[89]= 113

```

```

Out[90]= RandomGeneratorState[
  Method: ExtendedCA
  State hash: -5283743291589301093
]

```

```

In[91]:= list1 = RandomPrime[{46, 77}, 10]
Out[91]= {73, 67, 73, 61, 47, 47, 61, 59, 61, 73}

In[92]:= listFREQ = Tally[list1]
Out[92]= {{73, 3}, {67, 1}, {61, 3}, {47, 2}, {59, 1}}

In[93]:= listSORT = Sort[listFREQ[[All, 2]], Greater]
Out[93]= {3, 3, 2, 1, 1}

In[94]:= i = 1; While[listFREQ[[i, 2]] != listSORT[[1]], i++]
          maxfr = listFREQ[[i, 1]]
Out[95]= 73

In[96]:= u = 1; While[listFREQ[[u, 2]] != listSORT[[Length[listSORT]]], u++]
          minfr = listFREQ[[u, 1]]
Out[97]= 67

In[98]:= res1 = minfr * maxfr
Out[98]= 4891

```

**Найти значение функции Эйлера для числа x ,
 которое определяется из соотношения : $a * x + b = c \pmod{n}$,
 где $a = 34\,535$, $b = 34\,745$, $c = 26\,341$, $n = 11\,047$.**

Syntax:

"Найти значение функции Эйлера для числа x , которое определяется из соотношения : $a * x + b = c \pmod{n}$,
 где $a = 34\,535$, $b = 34\,745$, $c = 26\,341$, $n = 11\,047$." is incomplete; more input is needed.

```
In[99]:= a = 34 535
b = 34 745
c = 26 341
n = 11 047
i = 1; While[Mod[Mod[a, n] * i, n] ≠ Mod[c - b, n], i++]
Mod[Mod[a * i + b, n], n]
Mod[c, n]
EulerPhi[i]
```

```
Out[99]= 34 535
```

```
Out[100]=
34 745
```

```
Out[101]=
26 341
```

```
Out[102]=
11 047
```

```
Out[104]=
4247
```

```
Out[105]=
4247
```

```
Out[106]=
460
```

```
In[107]:= Определить ожидаемое время раскрытия пароля (tS)
длиной (S) 7 символов и содержащего следующие наборы :
{цифры, строчные русские, строчные латинские, прописные латинские},
если скорость перебора пароля (в символах в секунду) (R) равна обратному
элементу числа 1939 по модулю 661. Ответ вводить как целое число суток.
```

Syntax:

"Определить ожидаемое время раскрытия пароля (tS) длиной (S) 7 символов и содержащего следующие наборы : {цифры, строчные русские, строчные латинские, прописные латинские}, <<1>>" is incomplete; more input is needed.

```
In[10]:= S = 7
R = PowerMod[1939, -1, 661]
```

```
Out[10]= 7
```

```
Out[11]= 15
```



```

In[12]:= list1 = CharacterRange["0", "9"]
list2 = CharacterRange["а", "я"]
list3 = CharacterRange["a", "z"]
list4 = CharacterRange["A", "Z"]
list = Union[list1, list2, list3, list4]

Out[12]= {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Out[13]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н,
о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я}

Out[14]= {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}

Out[15]= {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}

Out[16]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю,
я, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, A, b, B, c, C, d, D, e, E, f, F, g, G, h, H, i, I, j, J,
k, K, l, L, m, M, n, N, o, O, p, P, q, Q, r, R, s, S, t, T, u, U, v, V, w, W, x, X, y, Y, z, Z}

In[17]:= A = Length[list]

Out[17]= 94

In[18]:= tS = IntegerPart[(1/2)*(A^S*1/R)/60/60/24]

Out[18]= 25 018 425

In[19]:= Архив текстового файла archive – 118. zip защищен паролем из 4 символов,
содержащих строчные и заглавные латинские буквы,
а также все цифры. Один из символов пароля можно определить из следующего
условия : полусумма кода символа и кода позиции символа в пароле равна 69,
полуразность кода символа и кода позиции символа равна 20.
Исключить пробелы и подсчитать число символов в тексте.

Syntax:
"Архив текстового файла archive – 118. zip защищен паролем из 4 символов, <<1>>, <<1>>, полуразность кода
символа и кода позиции символа равна 20. Исключить пробелы и подсчитать число символов в тексте." is

In[19]:= Установить генератор в нач состояние равным обратному
числа 1984 по модулю 199 Получить список сост из 100 случайных
строчных букв англ алфавита с нул начальными индексами . . .

Out[19]= 39 481 600 в с из по нач нул англ букв сост числа модулю
равным список Получить алфавита строчных генератор индексами
обратному случайных состояние Установить начальными . . .

```

```

In[20]:= param = PowerMod[1984, -1, 199]
SeedRandom[param]; Mas = RandomInteger[{1, 26}, 100];
engAlf = CharacterRange["a", "z"];
list1 = Table[0, 100];
For[i = 1, i < 101, i++, {list1[[i]] = engAlf[[Mas[[i]]]]}; list1
arr = Array[f, {10, 10}, 0];
Do[f[i, j] = list1[[10 * i + j + 1]], {i, 0, 9}, {j, 0, 9}]
TableForm[arr]
Print[arr[[3, 5]], arr[[3, 7]], arr[[6, 7]], arr[[6, 4]], arr[[9, 6]], arr[[9, 5]], arr[[8, 2]]

```

Out[20]= 33

```

Out[24]= {o, u, t, r, e, h, e, i, p, x, w, j, d, s, w, l, o, v, y, h, j, c, h, t,
j, e, u, v, r, w, f, o, v, y, d, y, e, n, e, w, p, i, m, v, j, g, g, i, x,
a, z, a, i, m, s, r, l, t, t, z, r, j, x, i, d, c, j, r, j, v, o, y, i, b,
b, z, q, z, z, j, q, h, v, p, c, q, u, d, f, r, n, q, n, z, r, d, b, w, s, x}

```

Out[27]//TableForm=

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| o | u | t | r | e | h | e | i | p | x |
| w | j | d | s | w | l | o | v | y | h |
| j | c | h | t | j | e | u | v | r | w |
| f | o | v | y | d | y | e | n | e | w |
| p | i | m | v | j | g | g | i | x | a |
| z | a | i | m | s | r | l | t | t | z |
| r | j | x | i | d | c | j | r | j | v |
| o | y | i | b | b | z | q | z | z | j |
| q | h | v | p | c | q | u | d | f | r |
| n | q | n | z | r | d | b | w | s | x |

julmqcy

```

In[29]:= Определить количество n во множестве ,
если при 30 экспериментах извлечения, коллизия возникает с
вероятностью 0.7. Ответ округлить до ближайшего большего целого

```

Syntax:

"Определить количество n во множестве, если при 30 экспериментах извлечения, коллизия возникает с вероятностью 0.7. Ответ округлить до ближайшего большего целого" is incomplete; more input is needed.

```

In[31]:= Solve[1 - E^((-30 * (30 - 1)) / (2 * n)) == 0.7, n]!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```

Out[31]= {}

In[32]:= **При стартовом значении ген случ чисел =
 29 сформировать последовательность состоящую
 из 892 случ цел чисел леж в диапазоне 195 –
 697. Найти произведение элементов последовательности принадлежащих
 подмножеству содержащему двойную коллизию. В поле для ответа ввести
 количество разрядов для 16 ричного представления полученного произведения**

Set: Tag Times in При ген случ чисел значении стартовом is Protected.

Out[32]= 5 044 260 в из леж цел случ чисел диапазоне состоящую сформировать последовательность –
 11 152. для² поле Найти ввести ответа двойную ричного разрядов элементов
 количество полученного содержащему подмножеству произведение
 произведения представления принадлежащих последовательности коллизию. В

```
In[33]:= SeedRandom[29]
mas = RandomChoice[Range[195, 697], 892]
a = 1
For[i = 195, i ≤ 697, i++, If[Count[mas, i] == 3, a = a*i,]]
Length[IntegerDigits[a, 16]]
```

```
Out[33]= RandomGeneratorState[
  Method: ExtendedCA
  State hash: -6131779296768577583
]
```

```
Out[34]= {386, 465, 644, 593, 491, 205, 486, 477, 266, 253, 199, 360, 281, 575, 555, 272, 528, 285,
  322, 274, 607, 418, 373, 539, 458, 464, 406, 677, 350, 604, 278, 212, 685, 530, 453,
  492, 539, 523, 446, 483, 382, 536, 635, 513, 258, 412, 439, 444, 461, 369, 282, 295,
  488, 273, 360, 484, 613, 620, 659, 657, 228, 228, 660, 210, 615, 204, 678, 543, 195,
  555, 413, 450, 577, 607, 411, 323, 638, 525, 655, 328, 405, 574, 558, 494, 597, 534,
  647, 354, 569, 443, 516, 431, 321, 495, 265, 515, 283, 304, 668, 273, 607, 647,
  686, 367, 462, 234, 504, 412, 543, 697, 350, 202, 519, 230, 329, 622, 297, 346,
  307, 469, 694, 646, 251, 599, 656, 539, 579, 249, 352, 565, 677, 608, 593, 548,
  291, 308, 195, 683, 541, 325, 577, 498, 518, 221, 225, 323, 647, 292, 566, 650,
  306, 333, 404, 609, 624, 493, 684, 549, 429, 690, 333, 229, 215, 392, 611, 403, 262,
  563, 645, 302, 433, 285, 649, 662, 325, 589, 321, 688, 420, 619, 525, 284, 574,
  625, 598, 549, 395, 449, 506, 229, 204, 354, 538, 411, 496, 564, 682, 553, 339,
  616, 310, 378, 586, 386, 304, 405, 228, 378, 693, 680, 652, 529, 462, 291, 413,
  602, 572, 451, 533, 501, 273, 258, 673, 497, 305, 292, 464, 635, 589, 326, 689, 653,
  337, 660, 636, 669, 231, 196, 369, 603, 271, 337, 316, 216, 513, 502, 229, 231,
  519, 443, 616, 443, 384, 685, 501, 249, 367, 261, 327, 488, 409, 473, 278, 271,
  448, 403, 637, 219, 475, 501, 552, 309, 416, 675, 650, 354, 566, 230, 330, 326, 352,
  455, 635, 292, 485, 465, 518, 305, 271, 396, 230, 585, 450, 429, 621, 391, 587,
  669, 226, 406, 476, 274, 359, 568, 432, 671, 286, 668, 203, 408, 620, 561, 508,
  204, 490, 420, 392, 210, 313, 326, 607, 298, 217, 483, 603, 664, 463, 552, 495, 599,
  367, 607, 519, 535, 288, 623, 529, 682, 697, 338, 218, 552, 613, 219, 413, 521,
  539, 282, 563, 653, 292, 578, 229, 263, 608, 582, 680, 643, 457, 202, 209, 670,
```

336, 460, 345, 549, 375, 310, 217, 433, 273, 477, 394, 548, 463, 205, 595, 495, 689,
 221, 316, 310, 495, 483, 612, 663, 606, 403, 271, 519, 614, 364, 398, 466, 339,
 681, 395, 622, 197, 259, 379, 692, 640, 522, 386, 234, 389, 360, 222, 313, 254,
 323, 414, 671, 354, 325, 197, 354, 401, 616, 363, 359, 228, 250, 492, 560, 510,
 453, 520, 439, 645, 637, 292, 274, 208, 415, 222, 493, 438, 310, 552, 446, 381, 682,
 646, 535, 501, 291, 651, 380, 667, 629, 348, 666, 344, 529, 583, 582, 549, 585,
 325, 478, 280, 433, 226, 232, 383, 446, 294, 406, 376, 681, 486, 268, 567, 534,
 469, 592, 689, 259, 392, 276, 573, 220, 498, 237, 531, 294, 442, 664, 241, 518,
 260, 350, 603, 421, 499, 305, 324, 403, 462, 626, 210, 254, 638, 393, 601, 611,
 306, 396, 592, 295, 694, 484, 491, 242, 351, 275, 415, 489, 256, 316, 257, 260,
 263, 208, 430, 581, 359, 448, 411, 380, 461, 221, 240, 316, 600, 326, 262, 288,
 558, 514, 544, 672, 605, 322, 256, 636, 583, 589, 514, 272, 562, 368, 666, 379,
 595, 420, 227, 433, 435, 686, 405, 667, 257, 293, 389, 374, 387, 396, 662, 677,
 494, 430, 416, 637, 653, 492, 600, 682, 212, 202, 495, 263, 653, 671, 240, 378,
 280, 403, 411, 242, 421, 265, 626, 648, 350, 593, 569, 315, 575, 426, 516, 356,
 355, 595, 340, 397, 685, 663, 580, 275, 322, 493, 592, 355, 606, 609, 418, 337,
 304, 266, 217, 561, 655, 278, 330, 333, 603, 435, 286, 364, 465, 694, 576, 479,
 657, 553, 360, 391, 637, 431, 448, 274, 241, 663, 649, 638, 200, 574, 576, 488,
 352, 510, 350, 367, 695, 365, 402, 668, 597, 632, 586, 409, 200, 527, 458, 264,
 298, 195, 398, 695, 406, 508, 381, 374, 519, 420, 540, 644, 241, 520, 263, 204,
 383, 498, 292, 665, 501, 669, 491, 201, 376, 478, 371, 417, 621, 453, 354, 697,
 415, 631, 641, 681, 527, 569, 388, 225, 646, 341, 370, 388, 509, 486, 691, 509,
 426, 500, 605, 650, 475, 654, 200, 249, 240, 484, 267, 401, 353, 418, 338, 371,
 400, 387, 638, 266, 600, 623, 644, 546, 511, 595, 230, 517, 365, 573, 470, 644,
 356, 238, 628, 200, 384, 665, 458, 325, 498, 674, 213, 512, 684, 474, 370, 527,
 658, 223, 627, 286, 437, 589, 290, 251, 276, 542, 433, 678, 594, 489, 643, 225,
 226, 599, 380, 333, 598, 498, 404, 296, 577, 263, 464, 603, 575, 204, 259, 657,
 382, 693, 430, 608, 630, 311, 637, 590, 593, 455, 203, 467, 655, 455, 616, 203,
 364, 467, 270, 686, 637, 624, 658, 448, 202, 311, 349, 213, 428, 671, 597, 424,
 228, 650, 281, 614, 653, 218, 697, 583, 673, 218, 349, 651, 577, 253, 608, 484,
 201, 593, 447, 507, 245, 513, 423, 226, 294, 196, 385, 480, 689, 542, 318, 644,
 359, 588, 201, 546, 247, 386, 554, 484, 693, 416, 657, 400, 649, 363, 627, 233,
 296, 217, 399, 445, 555, 437, 370, 271, 406, 570, 338, 364, 621, 305, 443, 474}

Out[35]= 1

Out[37]= 162

In[38]:= **В поле целых чисел определить сумму
 элементов приведенной системы вычетов по модулю 30**

Out[38]= 30 В по поле сумму целых чисел модулю
 вычетов системы элементов определить приведенной

```
In[39]:= poln = Range[0, 29]
Out[39]= {0, 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}
```

```
In[48]:= priv = {}
Out[48]= {}
```

```
In[49]:= privsum = 0
Out[49]= 0
```

```
In[50]:= For[i = 1, i < 30, i++,
            If[GCD[30, poln[[i]]] == 1, AppendTo[priv, poln[[i]]];
            privsum += poln[[i]]
          ]
Print[priv]
{1, 7, 11, 13, 17, 19, 23}
```

```
In[52]:= Print[privsum]
91
```

In[1]:= **В поле GF[313] определить произведение обратного элемента по сложению числа a = 241 и обратного элемента по умножению для числа b = 106**

Set: Tag Times in 241 и по для числа элемента обратного умножению b is Protected.

Set: Tag Times in В по поле числа сложению элемента обратного определить произведение a <<1>> is Protected.

```
Out[1]= 106
```

```
In[8]:= a5 = 241; b5 = 106; p5 = 313
k = 1
While[k < p5, If[Mod[b5 * k, p5] == 1, {Print[k];
Break;}, None];
k++]
```

```
Out[8]= 313
```

```
Out[9]= 1
251
```

```
In[11]:= Mod[106 * 251, 313]
```

```
Out[11]= 1
```

```
In[12]:= l = 1
Out[12]= 1
```

```
In[13]:= While[l < p5, If[Mod[a5 + l, p5] == 0, {Print[l]; Break;}, None]; l++]
72
```

```
In[14]:= Otvet = 251 * 72
```

```
Out[14]= 18 072
```

```
In[15]:= Определите кол – во положительных целых чисел,  
меньших 5045, которые взаимно просты с 5045.
```

Syntax: "Определите кол – во положительных целых чисел, меньших 5045, которые взаимно просты с 5045."
is incomplete; more input is needed.

```
In[15]:= a = 5045; k = 0
```

```
Do[If[GCD[a, i] == 1, k++], {i, a - 1, 0, -1}]; k
```

```
Out[15]= 0
```

```
Out[16]= 4032
```

```
In[17]:= a7 = 5045; k7 = 0
```

```
Do[If[GCD[a7, i] == 1, k7++, None], {i, 0, a7 - 1, 1}]  
k7
```

```
Out[17]= 0
```

```
Out[19]= 4032
```

```
In[20]:= Объединить 2 списка, перемешать, сумма позиций мин и макс
```

Syntax: "Объединить 2 списка, перемешать, сумма позиций мин и макс" is incomplete; more input is needed.

```
In[26]:= list3 = Join[Range[1, 70], Range[447, 1293]];
```

```
SeedRandom[869 582 105];
```

```
list4 = RandomSample[list3];
```

```
posmin = Position[list4, Min[list4]];
```

```
posmax = Position[list4, Max[list4]];
```

```
sump = posmax + posmin
```

```
Out[31]= {{563}}
```

```
In[32]:= Простые случайные числа, сумма последних 9
```

Syntax: "Простые случайные числа, сумма последних 9" is incomplete; more input is needed.

```
In[32]:= SeedRandom[935 213 969];
```

```
listP = RandomPrime[{6268, 31 340}, 26];
```

```
Sump = 0;
```

```
For[i = Length[listP] - 8, i ≤ Length[listP], i++, Sump = Sump + listP[[i]]
```

```
Sump
```

```
Out[35]= 175 391
```

```
In[36]:= Число простых чисел между
```

```
Out[36]= Число между чисел простых
```

```
In[37]:= PrimePi[29] - PrimePi[17]
```

```
Out[37]= 3
```

```
In[38]:= Установить генератор случайных чисел в начальное состояние с параметром  
603 018 939 (par). Получить список из 70 случайных целых чисел в диапазоне  
от 8839 до 80 019. Найти произведение последних пяти целых чисел.
```

Syntax: "чисел." is incomplete; more input is needed.

```
In[38]:= SeedRandom[603 018 939]
```

```
ListR = RandomInteger[{8839, 80 019}, 70]
```

```
Out[38]= RandomGeneratorState[  
Method: ExtendedCA  
State hash: 989 201 420 075 244 235  
]
```

```
Out[39]= {15 630, 67 392, 66 801, 78 232, 55 541, 31 592, 39 237, 75 363, 14 604, 74 762,  
66 347, 9840, 12 720, 55 804, 57 108, 14 145, 28 336, 65 570, 62 660, 11 197,  
26 681, 43 762, 23 491, 15 956, 72 311, 60 147, 21 456, 45 083, 59 080, 78 177,  
24 919, 9903, 11 637, 34 512, 47 488, 65 697, 79 636, 14 355, 60 290, 74 268,  
43 682, 58 496, 38 631, 24 049, 31 226, 22 479, 37 586, 26 929, 62 421, 30 884,  
14 946, 65 503, 35 515, 59 138, 79 772, 79 266, 57 862, 18 554, 49 963, 56 835,  
22 140, 45 090, 55 395, 60 487, 28 491, 11 520, 11 476, 23 229, 12 663, 63 081}
```

```
In[40]:= SumR = 0; For[i = 0, i < 5, i++, SumR += ListR[[70 - i]]]; SumR
```

```
Out[40]= 121 969
```

Архив текстового файла archive – 118. zip защищен паролем из 4 символов, содержащих строчные и заглавные латинские буквы, а также все цифры. Один из символов пароля можно определить из следующего условия : полусумма кода символа и кода позиции символа в пароле равна 69, полуразность кода символа и кода позиции символа равна 20.

Исключить пробелы и подсчитать число символов в тексте.

Алгоритм действий :

1. Решить систему линейных уравнений :

x – код символа , y – код позиции символа

$$(x + y) / 2 = 69$$

$$(x - y) / 2 = 20$$

$$x = 69 + 20 = 89$$

$$y = 2 * 69 - x = 49$$

Syntax:

"Архив текстового файла archive – 118. zip защищен паролем из 4 символов, содержащих строчные и заглавные латинские буквы, <<1>>, <<1>>-код символа, y-код позиции символа" is incomplete; more input is needed.

In[7]:=

FromCharacterCode[89]

FromCharacterCode[49]

Out[7]= Y

Out[8]= 1

In[9]:=

2. Запустить установленную программу AdvancedRARPasswordRecovery, ввести параметры (набор – латинские строчные, латинские заглавные, все цифры) и установить маску (Y ?? ?). Открыть архив archive-118.zip, подобрать пароль.

Syntax: Expression "(Y ?? ?). Открыть архив archive–118.zip, подобрать пароль." has no closing ")".

In[9]:=

3. Открыть текстовый файл из архива в Word. Скопировать текст в Mathematica, удалить пробелы, если необходимо.

Syntax:

"3. Открыть текстовый файл из архива в Word.Скопировать текст в Mathematica, удалить пробелы, если необходимо." is incomplete; more input is needed.


```
In[11]:= s1 = "fldggggggf"
          StringLength[s1]
```

```
Out[11]= fldggggggf
```

```
Out[12]= 10
```

In[1]:= **Скачайте с сетевого диска (ftp – сервера) файл text – 02. txt, расположенный в папке Texts и определите энтропию сообщения, содержащегося в нем . Ответ представить в битах , с 5 знаками после запятой .**

Так : (по идее в байтах)

Syntax:

"Скачайте с сетевого диска (ftp – сервера) файл text – 02. txt, расположенный в папке Texts и определите энтропию сообщения, содержащегося в <<1>> <<11>> в битах, с 5 знаками после запятой.Так : (по идее в байтах)" is incomplete; more input is needed.

textfile = нет

```
a = 23 728
```

```
b = 61 792
```

```
c = 27 326
```

```
n = 13 241
```

```
i = 1; While[Mod[Mod[a, n] * i, n] ≠ Mod[c - b, n], i++]
```

```
Mod[Mod[a * i + b, n], n]
```

```
Mod[c, n]
```

```
EulerPhi[i]
```

```
Out[9]= 23 728
```

```
Out[10]= 61 792
```

```
Out[11]= 27 326
```

```
Out[12]= 13 241
```

```
Out[14]= 844
```

```
Out[15]= 844
```

```
Out[16]= 1528
```

```

In[17]:= S = 9
         R = PowerMod[2036, -1, 677]

Out[17]= 9

Out[18]= 271

In[19]:= list2 = CharacterRange["а", "я"]
         list4 = CharacterRange["A", "Z"]
         list = Union[list2, list4]

Out[19]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н,
         о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я}

Out[20]= {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}

Out[21]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь,
         э, ю, я, 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}

In[22]:= A = Length[list]

Out[22]= 58

In[23]:= tS = IntegerPart[(1 / 2) * (A^S * 1 / R) / 60 / 60 / 24]

Out[23]= 158 613 048

In[49]:= par = PowerMod[2, 15, 179]
         SeedRandom[par]

Out[49]= 11

Out[50]= RandomGeneratorState[
  Method: ExtendedCA
  State hash: -3527631521982014537
]

In[51]:= list1 = RandomPrime[{26 000, 47 500}, 10 000]

Out[51]= {30 649, 46 093, 42 709, 40 751, 33 679, 41 903, 46 271, 42 407, 33 797,
         41 281, 45 389, 45 361, 42 293, 44 683, 43 633, 38 189, 37 501, 35 569,
         31 727, 41 203, 29 311, 43 933, 32 429, ... 9955 ..., 46 133, 42 773, 30 211,
         29 599, 46 591, 34 631, 43 633, 33 427, 31 151, 41 227, 47 051, 41 879,
         28 537, 41 947, 42 187, 35 251, 27 947, 28 151, 44 123, 29 803, 38 653, 38 959}

```

large output

[show less](#)[show more](#)[show all](#)[set size limit...](#)

```
In[52]:= listFREQ = Tally[list1]
```

Out[52]=

set size limit...

```
In[53]:= listSORT = Sort[listFREQ[[All, 2]], Greater]
```

[illegible]

[illegible]

```
In[54]:= i = 1; While[listFREQ[[i, 2]] ≠ listSORT[[1], i++]  
maxfr = listFREQ[[i, 1]]
```

Out[55]= 37 879

```
In[58]:= u = 1; While[listFREQ[u, 2] ≠ listSORT[Length[listSORT]], u++]
minfr = listFREQ[u, 1]
```

```
Out[59]= 44 563
```

```
res1 = minfr * maxfr
```

**Определить количество положительных чисел меньших а3,
которые взаимно просты с а3!!**

```
Out[60]= 1 688 001 877
```

```
In[9]:=
```

```
SistVichet[r_] := Module[{FVichet, PVichet, A1, A2},
  PVichet = {};
  FVichet = Range[0, r - 1];
  A1 = {FVichet, Length[FVichet]};
  For[i = 1, i ≤ Length[FVichet], i++,
  If[GCD[FVichet[[i]], r] == 1, AppendTo[PVichet, FVichet[[i]]]];
  A2 = {PVichet, Length[PVichet]};
  {A1, A2}]
```

```
In[10]:= a3 = 5045
```

```
SistVichet[a3][[2, 2]]
EulerPhi[a3]
```

```
Out[10]= 5045
```

```
Out[11]= 4032
```

```
Out[12]= 4032
```

```
In[71]:= Solve[1 - E^((-30 * (30 - 1)) / (2 * n)) == 0.7, n] нет
```

**Определить количество n во множестве,
если при а81 экспериментах извлечения, kr_2. nb 9 коллизия возникает
с вероятностью а82. Ответ округлить до ближайшего большего целого.**

Solve: 13241 is not a valid variable.

```
Out[71]= нет Solve[1 - 0.7 == e- $\frac{435}{*n}$ , 13 241]
```

Syntax:

"Определить количество n во множестве, если при а81 экспериментах извлечения, kr_2. nb 9 коллизия
возникает с вероятностью а82. Ответ округлить до ближайшего большего целого." is
incomplete; more input is needed.

```
In[67]:= a81 = 60
a82 = 0.4
Solve[1 - Exp[(-a81 * (a81 - 1)) / (2 * N2)] == a82, N2]
```

```
Out[67]= 60
```

```
Out[68]= 0.4
```

Solve: Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information.

```
Out[69]= {{N2 → 3464.98}}
```

```
In[70]:= Ceiling[3464.98]
```

```
Out[70]= 3465
```

```
In[117]:=
```

```
In[118]:=
```

```
In[119]:= PrimePi[6239] - PrimePi[282]
```

```
Out[119]=
751
```

Определить количество простых чисел, меньших или равных 7385

```
In[120]:= PrimePi[7385]
```

```
Out[120]=
938
```

```
2 799 908
```

```
Out[112]=
```

```
RandomGeneratorState[
  Method: ExtendedCA
  State hash: 4596086141809000869 ]
```

```
Out[113]=
```

```
{741, 499, 180, 653, 447, 230, 180, 342, 509, 648, 417, 344, 690, 427, 332, 172, 363, 625,
  695, 597, 382, 423, 288, 445, 752, 588, 357, 283, 513, 210, 449, 741, 545, 412, 689,
  559, 167, 166, 596, 361, 455, 584, 655, 625, 697, 360, 152, 754, 385, 369, 155, 561,
  603, 556, 467, 728, 675, 242, 597, 225, 380, 610, 170, 487, 644, 558, 624, 609, 635,
  237, 208, 334, 502, 353, 450, 637, 165, 553, 502, 641, 369, 329, 573, 187, 468, 197,
  599, 254, 439, 246, 215, 712, 554, 766, 761, 286, 434, 265, 291, 464, 553, 226, 292,
  436, 467, 486, 682, 323, 716, 542, 365, 415, 387, 512, 452, 188, 565, 508, 375,
  169, 719, 549, 238, 168, 733, 768, 575, 512, 602, 188, 294, 403, 764, 675, 426,
  532, 173, 381, 292, 782, 568, 737, 407, 200, 562, 338, 410, 439, 345, 235, 548,
  401, 290, 525, 560, 296, 765, 480, 707, 419, 720, 369, 380, 779, 611, 339, 310,
```

539, 333, 500, 300, 367, 627, 764, 413, 566, 408, 198, 719, 228, 619, 234, 494,
 748, 179, 574, 247, 327, 183, 457, 704, 554, 624, 451, 658, 484, 730, 691, 560,
 590, 719, 538, 773, 315, 520, 410, 528, 641, 767, 255, 467, 509, 408, 300, 733, 260,
 676, 246, 682, 739, 757, 660, 411, 245, 661, 708, 180, 261, 280, 596, 183, 781,
 523, 283, 324, 582, 726, 391, 736, 630, 357, 155, 767, 513, 750, 492, 404, 169,
 640, 495, 395, 219, 723, 285, 161, 241, 408, 324, 287, 343, 447, 503, 197, 474, 301,
 726, 674, 300, 197, 412, 751, 544, 450, 580, 589, 363, 295, 241, 247, 216, 565,
 184, 214, 750, 341, 350, 220, 514, 281, 238, 390, 480, 205, 443, 356, 148, 568,
 369, 505, 351, 336, 686, 458, 441, 684, 380, 778, 156, 201, 403, 551, 730, 504, 294,
 564, 778, 652, 659, 672, 320, 422, 457, 479, 468, 284, 408, 734, 204, 735, 327,
 463, 337, 538, 457, 781, 307, 364, 176, 286, 458, 749, 545, 295, 687, 265, 377,
 352, 180, 301, 184, 177, 548, 565, 501, 232, 152, 520, 489, 372, 516, 774, 377, 614,
 677, 527, 436, 492, 358, 671, 407, 178, 616, 714, 293, 214, 362, 659, 500, 153,
 461, 691, 365, 161, 393, 422, 650, 495, 390, 578, 351, 322, 233, 234, 727, 150,
 437, 416, 635, 442, 359, 598, 224, 273, 317, 343, 487, 691, 511, 640, 573, 443,
 316, 767, 308, 671, 496, 420, 459, 246, 201, 485, 306, 265, 626, 477, 505, 268, 493,
 591, 359, 150, 249, 615, 549, 710, 157, 494, 446, 626, 510, 560, 682, 777, 295,
 430, 385, 535, 342, 566, 474, 279, 446, 485, 249, 720, 687, 674, 420, 261, 366,
 609, 520, 685, 302, 324, 284, 774, 232, 162, 301, 423, 205, 335, 380, 589, 401,
 369, 667, 582, 455, 250, 765, 354, 570, 180, 707, 548, 748, 566, 493, 705, 306, 425,
 161, 436, 762, 299, 390, 653, 635, 612, 517, 313, 656, 171, 456, 295, 359, 423,
 577, 246, 242, 395, 554, 421, 347, 192, 195, 398, 256, 358, 257, 148, 717, 775,
 666, 769, 185, 152, 287, 312, 696, 554, 704, 532, 737, 285, 173, 591, 738, 318, 508,
 293, 352, 162, 542, 690, 193, 664, 536, 564, 769, 388, 532, 309, 616, 773, 322,
 714, 486, 200, 619, 595, 506, 333, 499, 234, 647, 717, 475, 280, 411, 266, 274,
 316, 549, 556, 403, 230, 651, 211, 701, 652, 756, 661, 423, 673, 745, 623, 733, 582,
 605, 322, 612, 555, 435, 487, 500, 741, 761, 164, 220, 250, 682, 757, 682, 582,
 288, 304, 550, 775, 504, 646, 239, 158, 171, 237, 719, 687, 511, 210, 516, 236,
 456, 494, 215, 397, 519, 594, 395, 639, 681, 241, 318, 250, 436, 187, 253, 288,
 520, 397, 149, 643, 404, 700, 706, 594, 166, 488, 711, 672, 539, 248, 381, 293,
 524, 170, 660, 336, 550, 759, 281, 235, 246, 600, 246, 406, 313, 313, 573, 392,
 616, 429, 640, 258, 387, 198, 342, 360, 494, 523, 478, 770, 643, 154, 677, 623,
 170, 749, 746, 220, 533, 271, 738, 426, 556, 751, 474, 561, 151, 150, 394, 390,
 631, 660, 289, 228, 450, 297, 263, 681, 701, 479, 517, 688, 755, 408, 361, 724,
 749, 434, 502, 258, 556, 549, 708, 194, 322, 584, 768, 165, 380, 768, 403, 447,
 503, 218, 309, 341, 366, 157, 555, 206, 769, 366, 696, 641, 614, 414, 324, 151,
 479, 338, 177, 150, 464, 252, 656, 646, 547, 777, 602, 625, 554, 229, 266, 251,
 247, 596, 746, 424, 161, 156, 693, 708, 205, 312, 571, 420, 556, 149, 174, 304,
 485, 765, 313, 151, 467, 677, 503, 211, 364, 165, 227, 352, 617, 352, 584, 747,
 618, 449, 219, 407, 638, 203, 556, 394, 617, 544, 197, 693, 693, 165, 708, 557,
 758, 410, 688, 354, 733, 776, 695, 342, 737, 754, 532, 542, 665, 753, 556, 613,
 582, 187, 687, 423, 588, 417, 429, 221, 339, 237, 353, 779, 753, 322, 274, 525,
 536, 508, 385, 451, 598, 201, 618, 782, 299, 454, 729, 476, 202, 535, 406, 721, 440}

```
Out[114]=
```

```
1
```

```
Out[116]=
```

```
129
```



```

In[*]:= SistVichet[r_] := Module[{FVichet, PVichet, A1, A2},
    PVichet = {};
    FVichet = Range[0, r - 1];
    A1 = {FVichet, Length[FVichet]};
    For[i = 1, i ≤ Length[FVichet], i++,
        If[GCD[FVichet[[i]], r] == 1, AppendTo[PVichet, FVichet[[i]]]];
    A2 = {PVichet, Length[PVichet]};
    {A1, A2}]

```

Определить обратный элемент числа a1, в поле GF(b1)

```

In[*]:= a1 = 33 307
        b1 = 647 371

```

```
Out[*]= 33 307
```

```
Out[*]= 647 371
```

```

In[*]:= PowerMod[a1, -1, b1]

```

```
Out[*]= 300 352
```

В поле целых чисел определить сумму элементов приведенной системы вычетов по модулю a2 (если a2 простое, то все как здесь, иначе хз)

```

a2 = 2207
PrimeQ[a2]
Sum[i, {i, 1, a2 - 1}]

```

```
Out[*]= 2207
```

```
Out[*]= True
```

```
Out[*]= 2 434 321
```

Общий случай

```

In[*]:= list2 = SistVichet[2207][[2]]

```

```

Out[*]= { {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, 42, 43, 44, 45, 46, 47,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108,
109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125,
126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159,
160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176,
177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193,
194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210,
211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227,

```

228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244,
245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261,
262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278,
279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295,
296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312,
313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329,
330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346,
347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380,
381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397,
398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414,
415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431,
432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448,
449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465,
466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482,
483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499,
500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516,
517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550,
551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567,
568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584,
585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601,
602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618,
619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635,
636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652,
653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669,
670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686,
687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703,
704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720,
721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737,
738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754,
755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771,
772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788,
789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805,
806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822,
823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839,
840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856,
857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873,
874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890,
891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907,
908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924,
925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941,
942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958,
959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974,
975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990,
991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005,
1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019,
1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033,
1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047,
1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061,
1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075,
1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089,
1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103,
1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117,
1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131,

| | | | | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| 1132 | 1133 | 1134 | 1135 | 1136 | 1137 | 1138 | 1139 | 1140 | 1141 | 1142 | 1143 | 1144 | 1145 |
| 1146 | 1147 | 1148 | 1149 | 1150 | 1151 | 1152 | 1153 | 1154 | 1155 | 1156 | 1157 | 1158 | 1159 |
| 1160 | 1161 | 1162 | 1163 | 1164 | 1165 | 1166 | 1167 | 1168 | 1169 | 1170 | 1171 | 1172 | 1173 |
| 1174 | 1175 | 1176 | 1177 | 1178 | 1179 | 1180 | 1181 | 1182 | 1183 | 1184 | 1185 | 1186 | 1187 |
| 1188 | 1189 | 1190 | 1191 | 1192 | 1193 | 1194 | 1195 | 1196 | 1197 | 1198 | 1199 | 1200 | 1201 |
| 1202 | 1203 | 1204 | 1205 | 1206 | 1207 | 1208 | 1209 | 1210 | 1211 | 1212 | 1213 | 1214 | 1215 |
| 1216 | 1217 | 1218 | 1219 | 1220 | 1221 | 1222 | 1223 | 1224 | 1225 | 1226 | 1227 | 1228 | 1229 |
| 1230 | 1231 | 1232 | 1233 | 1234 | 1235 | 1236 | 1237 | 1238 | 1239 | 1240 | 1241 | 1242 | 1243 |
| 1244 | 1245 | 1246 | 1247 | 1248 | 1249 | 1250 | 1251 | 1252 | 1253 | 1254 | 1255 | 1256 | 1257 |
| 1258 | 1259 | 1260 | 1261 | 1262 | 1263 | 1264 | 1265 | 1266 | 1267 | 1268 | 1269 | 1270 | 1271 |
| 1272 | 1273 | 1274 | 1275 | 1276 | 1277 | 1278 | 1279 | 1280 | 1281 | 1282 | 1283 | 1284 | 1285 |
| 1286 | 1287 | 1288 | 1289 | 1290 | 1291 | 1292 | 1293 | 1294 | 1295 | 1296 | 1297 | 1298 | 1299 |
| 1300 | 1301 | 1302 | 1303 | 1304 | 1305 | 1306 | 1307 | 1308 | 1309 | 1310 | 1311 | 1312 | 1313 |
| 1314 | 1315 | 1316 | 1317 | 1318 | 1319 | 1320 | 1321 | 1322 | 1323 | 1324 | 1325 | 1326 | 1327 |
| 1328 | 1329 | 1330 | 1331 | 1332 | 1333 | 1334 | 1335 | 1336 | 1337 | 1338 | 1339 | 1340 | 1341 |
| 1342 | 1343 | 1344 | 1345 | 1346 | 1347 | 1348 | 1349 | 1350 | 1351 | 1352 | 1353 | 1354 | 1355 |
| 1356 | 1357 | 1358 | 1359 | 1360 | 1361 | 1362 | 1363 | 1364 | 1365 | 1366 | 1367 | 1368 | 1369 |
| 1370 | 1371 | 1372 | 1373 | 1374 | 1375 | 1376 | 1377 | 1378 | 1379 | 1380 | 1381 | 1382 | 1383 |
| 1384 | 1385 | 1386 | 1387 | 1388 | 1389 | 1390 | 1391 | 1392 | 1393 | 1394 | 1395 | 1396 | 1397 |
| 1398 | 1399 | 1400 | 1401 | 1402 | 1403 | 1404 | 1405 | 1406 | 1407 | 1408 | 1409 | 1410 | 1411 |
| 1412 | 1413 | 1414 | 1415 | 1416 | 1417 | 1418 | 1419 | 1420 | 1421 | 1422 | 1423 | 1424 | 1425 |
| 1426 | 1427 | 1428 | 1429 | 1430 | 1431 | 1432 | 1433 | 1434 | 1435 | 1436 | 1437 | 1438 | 1439 |
| 1440 | 1441 | 1442 | 1443 | 1444 | 1445 | 1446 | 1447 | 1448 | 1449 | 1450 | 1451 | 1452 | 1453 |
| 1454 | 1455 | 1456 | 1457 | 1458 | 1459 | 1460 | 1461 | 1462 | 1463 | 1464 | 1465 | 1466 | 1467 |
| 1468 | 1469 | 1470 | 1471 | 1472 | 1473 | 1474 | 1475 | 1476 | 1477 | 1478 | 1479 | 1480 | 1481 |
| 1482 | 1483 | 1484 | 1485 | 1486 | 1487 | 1488 | 1489 | 1490 | 1491 | 1492 | 1493 | 1494 | 1495 |
| 1496 | 1497 | 1498 | 1499 | 1500 | 1501 | 1502 | 1503 | 1504 | 1505 | 1506 | 1507 | 1508 | 1509 |
| 1510 | 1511 | 1512 | 1513 | 1514 | 1515 | 1516 | 1517 | 1518 | 1519 | 1520 | 1521 | 1522 | 1523 |
| 1524 | 1525 | 1526 | 1527 | 1528 | 1529 | 1530 | 1531 | 1532 | 1533 | 1534 | 1535 | 1536 | 1537 |
| 1538 | 1539 | 1540 | 1541 | 1542 | 1543 | 1544 | 1545 | 1546 | 1547 | 1548 | 1549 | 1550 | 1551 |
| 1552 | 1553 | 1554 | 1555 | 1556 | 1557 | 1558 | 1559 | 1560 | 1561 | 1562 | 1563 | 1564 | 1565 |
| 1566 | 1567 | 1568 | 1569 | 1570 | 1571 | 1572 | | | | | | | |

```

1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915,
1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929,
1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943,
1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957,
1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971,
1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985,
1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013,
2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027,
2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041,
2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055,
2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069,
2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083,
2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097,
2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111,
2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125,
2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139,
2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153,
2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167,
2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181,
2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194,
2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206}, 2206}

```

```
In[ ]:= Sum[list2[[1, i]], {i, 1, list2[[2]]}]
```

сумма

```
Out[ ]:= 2 434 321
```

Определить количество положительных чисел меньших a_3 , которые взаимно просты с a_3

```
In[ ]:= a3 = 2184
```

```
SistVichet[a3][[2, 2]]
```

```
EulerPhi[a3]
```

функция Эйлера

```
Out[ ]:= 2184
```

```
Out[ ]:= 576
```

```
Out[ ]:= 576
```

Скачайте с сетевого диска (ftp-сервера) файл ..., расположенный в папке ... и определите энтропию сообщения, содержащегося в нем. Ответ представить в битах, с 7 знаками после запятой. Пример ввода 1.1111111

```
In[*]:= v27 = ReadList["D:\\Dekstop\\spotifyapi.txt", Byte]
```

[считать в ...](#) [дифференцировать](#) [байт](#)

```
Out[*]:= {99, 108, 105, 101, 110, 116, 32, 105, 100, 58, 32, 97, 52, 56, 52, 56, 55, 100, 97, 56,
53, 98, 55, 52, 100, 99, 53, 98, 56, 52, 100, 50, 57, 102, 102, 101, 51, 50, 97, 99,
99, 49, 50, 10, 99, 108, 105, 101, 110, 116, 32, 115, 101, 99, 114, 101, 116, 58,
32, 100, 102, 52, 56, 51, 49, 49, 48, 54, 102, 101, 55, 52, 99, 54, 48, 98, 97, 98,
49, 51, 98, 57, 102, 48, 100, 48, 50, 99, 99, 102, 51, 10, 232, 236, 255, 32, 239,
238, 235, 252, 231, 238, 226, 224, 242, 229, 235, 255, 58, 32, 51, 49, 114, 110,
119, 104, 120, 107, 121, 51, 54, 109, 99, 112, 115, 103, 99, 52, 122, 100, 119, 112,
54, 120, 104, 104, 55, 113, 10, 10, 240, 254, 32, 225, 253, 240, 240, 232, 236, 238,
240, 58, 32, 102, 109, 48, 106, 101, 54, 100, 117, 51, 51, 115, 48, 101, 49, 121,
122, 122, 107, 114, 106, 104, 116, 113, 114, 114, 10, 10, 10, 104, 116, 116, 112,
115, 58, 47, 47, 101, 120, 97, 109, 112, 108, 101, 46, 99, 111, 109, 47, 99, 97,
108, 108, 98, 97, 99, 107, 63, 99, 111, 100, 101, 61, 65, 81, 68, 90, 74, 120, 79,
72, 78, 66, 103, 83, 88, 95, 107, 70, 117, 48, 90, 48, 74, 120, 98, 109, 72, 88, 65,
86, 107, 71, 113, 77, 119, 101, 85, 100, 56, 54, 88, 112, 105, 103, 72, 98, 97, 108,
54, 79, 95, 73, 102, 84, 90, 89, 107, 75, 101, 116, 78, 79, 90, 118, 50, 75, 106, 73,
98, 118, 114, 85, 100, 51, 54, 106, 85, 89, 112, 78, 105, 105, 89, 109, 104, 102,
121, 114, 102, 119, 55, 52, 84, 95, 117, 71, 70, 102, 78, 90, 113, 73, 78, 73, 53,
48, 73, 55, 69, 68, 95, 102, 90, 54, 99, 117, 48, 121, 67, 108, 111, 121, 76, 95,
76, 56, 89, 81, 117, 113, 109, 111, 121, 97, 52, 98, 57, 57, 81, 111, 51, 76, 101,
100, 77, 98, 51, 100, 78, 118, 86, 50, 101, 98, 45, 116, 104, 117, 72, 97, 82, 100,
54, 104, 121, 83, 101, 104, 102, 119, 114, 97, 97, 48, 89, 49, 108, 97, 72, 104, 56,
66, 80, 86, 70, 82, 102, 76, 49, 98, 112, 57, 108, 88, 69, 67, 71, 70, 84, 67, 52,
74, 117, 79, 108, 48, 108, 99, 89, 73, 75, 112, 104, 107, 69, 105, 102, 117, 99, 78,
69, 38, 115, 116, 97, 116, 101, 61, 51, 52, 102, 70, 115, 50, 57, 107, 100, 48, 57}
```

```
In[*]:= freq27 = Tally[v27]
```

[подсчитать](#)

```
Out[*]:= {{99, 18}, {108, 11}, {105, 7}, {101, 17}, {110, 3}, {116, 10}, {32, 8}, {100, 15},
{58, 5}, {97, 14}, {52, 11}, {56, 8}, {55, 6}, {53, 3}, {98, 13}, {50, 7}, {57, 7},
{102, 16}, {51, 12}, {49, 8}, {10, 7}, {115, 6}, {114, 8}, {48, 13}, {54, 10}, {232, 2},
{236, 2}, {255, 2}, {239, 1}, {238, 3}, {235, 2}, {252, 1}, {231, 1}, {226, 1}, {224, 1},
{242, 1}, {229, 1}, {119, 5}, {104, 11}, {120, 5}, {107, 8}, {121, 7}, {109, 7},
{112, 8}, {103, 3}, {122, 3}, {113, 5}, {240, 4}, {254, 1}, {225, 1}, {253, 1},
{106, 4}, {117, 8}, {47, 3}, {46, 1}, {111, 5}, {63, 1}, {61, 2}, {65, 2}, {81, 3},
{68, 2}, {90, 6}, {74, 3}, {79, 4}, {72, 5}, {78, 7}, {66, 2}, {83, 2}, {88, 4},
{95, 5}, {70, 5}, {86, 3}, {71, 3}, {77, 2}, {85, 3}, {73, 6}, {84, 3}, {89, 6},
{75, 3}, {118, 3}, {69, 4}, {67, 3}, {76, 4}, {45, 1}, {82, 2}, {80, 1}, {38, 1}}
```

```
In[ ]:= p27 = N[freq27[[A11, 2]] / Length[v27]]
```

```
Out[ ]:= {0.039823, 0.0243363, 0.0154867, 0.0376106, 0.00663717, 0.0221239, 0.0176991,
0.0331858, 0.0110619, 0.0309735, 0.0243363, 0.0176991, 0.0132743, 0.00663717,
0.0287611, 0.0154867, 0.0154867, 0.0353982, 0.0265487, 0.0176991, 0.0154867,
0.0132743, 0.0176991, 0.0287611, 0.0221239, 0.00442478, 0.00442478, 0.00442478,
0.00221239, 0.00663717, 0.00442478, 0.00221239, 0.00221239, 0.00221239,
0.00221239, 0.00221239, 0.00221239, 0.0110619, 0.0243363, 0.0110619, 0.0176991,
0.0154867, 0.0154867, 0.0176991, 0.00663717, 0.00663717, 0.0110619, 0.00884956,
0.00221239, 0.00221239, 0.00221239, 0.00884956, 0.0176991, 0.00663717,
0.00221239, 0.0110619, 0.00221239, 0.00442478, 0.00442478, 0.00663717,
0.00442478, 0.0132743, 0.00663717, 0.00884956, 0.0110619, 0.0154867, 0.00442478,
0.00442478, 0.00884956, 0.0110619, 0.0110619, 0.00663717, 0.00663717, 0.00442478,
0.00663717, 0.0132743, 0.00663717, 0.0132743, 0.00663717, 0.00663717, 0.00884956,
0.00663717, 0.00884956, 0.00221239, 0.00442478, 0.00221239, 0.00221239}
```

```
In[ ]:= np27 = Length[p27]
```

```
Out[ ]:= 87
```

```
In[ ]:= summp27 = Sum[p27[[i]], {i, np27}]
```

```
Out[ ]:= 1.
```

```
In[ ]:= ent27 = -Sum[p27[[i]] * Log[2, p27[[i]]], {i, np27}]
```

```
Out[ ]:= 6.0379
```

```
In[ ]:= NumberForm[ent27, 8]
```

Последний нолик не вывелся

```
Out[ ]//NumberForm=
6.037895
```

Установить генератор случайных чисел в начальное состояние с параметром, равным обратному элементу числа a51 по модулю a52. Получить список, состоящий из 100 случайных строчных букв английского алфавита. Инициализировать массив 10*10 с нулевыми начальными индексами элементами этого списка. Преобразовать элементы массива с индексами ... в строку и ввести в поле ввода.

```

In[ ]:= a51 = 1042
a52 = 109
SeedRandom[PowerMod[a51, -1, a52]]
инициализация степень по модулю
Array5 = Array[mass5, {10, 10}, {0, 0}]
массив

list5 = Partition[RandomChoice[CharacterRange["a", "z"], 100], 10]
разбиение случайный выбор ряд символов

Do[mass5[i, j] = list5[[i + 1, j + 1]], {i, 0, 9}, {j, 0, 9}]
оператор цикла

Array5 // TableForm
табличная форма

StringJoin[{mass5[2, 2], mass5[6, 5],
соединить строки
  mass5[9, 3], mass5[5, 2], mass5[7, 3], mass5[3, 7], mass5[3, 3]}]

```

Out[]:= 1042

Out[]:= 109

```

Out[ ]:= { {mass5[0, 0], mass5[0, 1], mass5[0, 2], mass5[0, 3], mass5[0, 4],
  mass5[0, 5], mass5[0, 6], mass5[0, 7], mass5[0, 8], mass5[0, 9]},
  {mass5[1, 0], mass5[1, 1], mass5[1, 2], mass5[1, 3], mass5[1, 4],
  mass5[1, 5], mass5[1, 6], mass5[1, 7], mass5[1, 8], mass5[1, 9]},
  {mass5[2, 0], mass5[2, 1], mass5[2, 2], mass5[2, 3], mass5[2, 4],
  mass5[2, 5], mass5[2, 6], mass5[2, 7], mass5[2, 8], mass5[2, 9]},
  {mass5[3, 0], mass5[3, 1], mass5[3, 2], mass5[3, 3], mass5[3, 4],
  mass5[3, 5], mass5[3, 6], mass5[3, 7], mass5[3, 8], mass5[3, 9]},
  {mass5[4, 0], mass5[4, 1], mass5[4, 2], mass5[4, 3], mass5[4, 4],
  mass5[4, 5], mass5[4, 6], mass5[4, 7], mass5[4, 8], mass5[4, 9]},
  {mass5[5, 0], mass5[5, 1], mass5[5, 2], mass5[5, 3], mass5[5, 4],
  mass5[5, 5], mass5[5, 6], mass5[5, 7], mass5[5, 8], mass5[5, 9]},
  {mass5[6, 0], mass5[6, 1], mass5[6, 2], mass5[6, 3], mass5[6, 4],
  mass5[6, 5], mass5[6, 6], mass5[6, 7], mass5[6, 8], mass5[6, 9]},
  {mass5[7, 0], mass5[7, 1], mass5[7, 2], mass5[7, 3], mass5[7, 4],
  mass5[7, 5], mass5[7, 6], mass5[7, 7], mass5[7, 8], mass5[7, 9]},
  {mass5[8, 0], mass5[8, 1], mass5[8, 2], mass5[8, 3], mass5[8, 4],
  mass5[8, 5], mass5[8, 6], mass5[8, 7], mass5[8, 8], mass5[8, 9]},
  {mass5[9, 0], mass5[9, 1], mass5[9, 2], mass5[9, 3], mass5[9, 4],
  mass5[9, 5], mass5[9, 6], mass5[9, 7], mass5[9, 8], mass5[9, 9]} }

```

```

Out[ ]:= { {z, t, x, y, z, l, q, y, u, l}, {i, c, f, e, s, z, u, j, q, f},
  {o, q, d, o, p, o, j, c, s, l}, {d, y, m, r, m, o, x, a, z, h}, {t, u, t, m, h, e, o, t, b, y},
  {i, f, o, u, v, a, h, n, e, b}, {l, v, o, h, f, g, s, o, q, k}, {g, l, w, f, b, t, b, m, l, m},
  {r, j, f, o, a, v, z, j, i, b}, {r, f, m, n, o, d, q, d, n, c} }

```

Out[]:= TableForm=

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| z | t | x | y | z | l | q | y | u | l |
| i | c | f | e | s | z | u | j | q | f |
| o | q | d | o | p | o | j | c | s | l |
| d | y | m | r | m | o | x | a | z | h |
| t | u | t | m | h | e | o | t | b | y |
| i | f | o | u | v | a | h | n | e | b |
| l | v | o | h | f | g | s | o | q | k |
| g | l | w | f | b | t | b | m | l | m |
| r | j | f | o | a | v | z | j | i | b |
| r | f | m | n | o | d | q | d | n | c |

Out[]:= dgnofar

В поле GF[a6] определить произведение обратного элемента по сложению для числа a61 и обратного элемента по умножению для числа a62.

```
In[ ]:= a6 = 197
        a61 = 8
        a62 = 36

Out[ ]:= 197

Out[ ]:= 8

Out[ ]:= 36

In[ ]:= (a6 - a61) * PowerMod[a62, -1, a6]
        |степень по модулю

Out[ ]:= 19 656
```

При стартовом значении генератора случайных чисел равном a71 сформировать последовательность, состоящую из a72 случайных целых чисел, лежащих в диапазоне [a73, a74]. Найти произведение элементов последовательности, принадлежащих подмножеству, содержащему двойную коллизия. В поле для ответа ввести количество разрядов для двоичного представления полученного произведения.

```
In[ ]:= a71 = 45
        a72 = 853
        a73 = 179
        a74 = 651
        SeedRandom[a71]
        |инициализация генератора псевдослучайных чисел
        list71 = Tally[RandomInteger[{a73, a74}, a72]]
        |подс... |случайное целое число

        answ7 = 1
        Do[If[list71[[i, 2]] == 3, answ7 = answ7 * list71[[i, 1]]], {i, 1, Length[list71]}]
        |... |условный оператор |длина
        IntegerLength[answ7, 2]
        |длина целого числа

Out[ ]:= 45

Out[ ]:= 853

Out[ ]:= 179

Out[ ]:= 651
```



```

Out[ ]= {{476, 3}, {538, 3}, {267, 1}, {432, 3}, {593, 1}, {389, 5}, {289, 2}, {650, 3}, {315, 5},
{633, 3}, {247, 4}, {395, 4}, {299, 3}, {229, 3}, {352, 2}, {573, 4}, {454, 2}, {297, 4},
{578, 3}, {400, 2}, {248, 2}, {529, 3}, {642, 4}, {348, 1}, {368, 5}, {380, 5}, {182, 2},
{197, 4}, {490, 2}, {586, 3}, {321, 3}, {437, 2}, {553, 2}, {381, 3}, {336, 1}, {439, 1},
{266, 1}, {631, 3}, {466, 2}, {384, 1}, {430, 1}, {604, 1}, {469, 3}, {362, 1},
{224, 1}, {406, 3}, {339, 3}, {265, 5}, {281, 3}, {253, 2}, {570, 3}, {545, 3},
{513, 1}, {250, 1}, {311, 1}, {298, 3}, {504, 1}, {252, 3}, {392, 3}, {249, 2},
{623, 2}, {330, 3}, {528, 2}, {232, 2}, {193, 3}, {328, 1}, {607, 4}, {620, 2},
{615, 3}, {498, 3}, {644, 3}, {519, 1}, {632, 1}, {501, 2}, {541, 2}, {438, 3},
{581, 4}, {358, 1}, {270, 4}, {640, 2}, {546, 2}, {309, 2}, {638, 2}, {397, 1}, {429, 2},
{279, 2}, {337, 1}, {223, 1}, {222, 3}, {480, 1}, {618, 3}, {568, 3}, {461, 2},
{319, 3}, {325, 1}, {565, 2}, {350, 4}, {416, 2}, {532, 2}, {254, 2}, {366, 1},
{574, 2}, {452, 4}, {509, 2}, {221, 2}, {534, 2}, {495, 1}, {514, 2}, {285, 4},
{376, 2}, {214, 2}, {527, 5}, {345, 3}, {396, 2}, {393, 3}, {333, 3}, {624, 1},
{477, 2}, {453, 1}, {591, 1}, {189, 4}, {410, 3}, {536, 4}, {614, 2}, {185, 2},
{451, 2}, {494, 2}, {207, 3}, {373, 3}, {388, 3}, {310, 3}, {346, 1}, {444, 2},
{255, 1}, {585, 3}, {365, 2}, {418, 4}, {556, 3}, {471, 3}, {436, 4}, {608, 4},
{611, 1}, {470, 2}, {500, 3}, {280, 2}, {520, 2}, {331, 4}, {385, 5}, {419, 2},
{241, 3}, {539, 4}, {359, 2}, {448, 2}, {483, 2}, {443, 1}, {198, 4}, {577, 2},
{338, 1}, {424, 1}, {308, 1}, {374, 1}, {521, 3}, {353, 2}, {508, 2}, {403, 1},
{530, 2}, {199, 3}, {505, 1}, {268, 1}, {554, 3}, {525, 1}, {564, 1}, {540, 2}, {605, 1},
{445, 5}, {516, 1}, {286, 3}, {179, 3}, {203, 2}, {463, 2}, {220, 3}, {422, 2},
{499, 2}, {235, 2}, {372, 1}, {317, 3}, {269, 1}, {379, 3}, {407, 4}, {409, 2},
{413, 2}, {580, 1}, {320, 1}, {360, 2}, {464, 3}, {601, 1}, {239, 1}, {293, 3},
{347, 2}, {227, 3}, {610, 3}, {433, 2}, {200, 1}, {212, 3}, {588, 3}, {398, 3},
{507, 1}, {478, 5}, {357, 4}, {408, 5}, {550, 3}, {192, 5}, {334, 1}, {411, 3},
{341, 2}, {213, 1}, {511, 3}, {240, 2}, {187, 2}, {549, 4}, {458, 2}, {186, 2},
{457, 1}, {435, 4}, {450, 2}, {459, 2}, {329, 2}, {313, 2}, {387, 1}, {238, 6},
{401, 1}, {287, 1}, {342, 1}, {283, 2}, {485, 1}, {484, 1}, {327, 1}, {399, 1},
{594, 2}, {304, 5}, {613, 2}, {456, 2}, {576, 1}, {335, 3}, {205, 4}, {180, 2},
{211, 2}, {506, 1}, {323, 2}, {518, 1}, {548, 1}, {472, 4}, {537, 2}, {314, 1},
{275, 2}, {303, 1}, {282, 2}, {259, 1}, {306, 4}, {598, 1}, {559, 1}, {474, 2},
{231, 2}, {256, 2}, {597, 1}, {488, 3}, {386, 1}, {420, 1}, {361, 3}, {491, 2},
{582, 2}, {261, 1}, {547, 1}, {473, 1}, {589, 4}, {242, 2}, {434, 1}, {447, 1}, {295, 2},
{344, 5}, {307, 1}, {645, 3}, {427, 4}, {569, 1}, {322, 1}, {260, 3}, {300, 2},
{369, 1}, {264, 1}, {542, 1}, {294, 2}, {217, 1}, {183, 2}, {405, 1}, {417, 1},
{290, 1}, {188, 5}, {479, 1}, {449, 3}, {489, 1}, {524, 1}, {600, 1}, {390, 1},
{627, 1}, {370, 3}, {523, 2}, {592, 3}, {210, 3}, {551, 1}, {575, 2}, {602, 3},
{262, 3}, {194, 2}, {412, 1}, {258, 2}, {616, 3}, {442, 1}, {446, 2}, {440, 3},
{421, 2}, {195, 2}, {355, 3}, {237, 3}, {630, 2}, {535, 1}, {648, 1}, {426, 3},
{236, 1}, {230, 1}, {584, 3}, {271, 1}, {609, 3}, {215, 1}, {441, 4}, {562, 3},
{497, 2}, {305, 1}, {617, 1}, {382, 3}, {225, 2}, {552, 1}, {606, 4}, {292, 1},
{561, 1}, {402, 3}, {572, 1}, {318, 2}, {651, 2}, {579, 1}, {228, 3}, {363, 2},
{583, 1}, {557, 3}, {414, 2}, {467, 2}, {216, 1}, {460, 3}, {301, 2}, {533, 1},
{626, 1}, {531, 1}, {196, 1}, {563, 1}, {465, 2}, {246, 1}, {274, 2}, {190, 2},
{226, 1}, {515, 1}, {404, 1}, {629, 1}, {340, 2}, {481, 1}, {302, 2}, {277, 1},
{455, 1}, {641, 1}, {272, 1}, {184, 1}, {191, 1}, {487, 1}, {371, 1}, {296, 1},
{544, 1}, {596, 1}, {639, 1}, {263, 1}, {202, 1}, {468, 1}, {512, 1}, {204, 1},
{243, 2}, {425, 1}, {364, 1}, {208, 1}, {326, 2}, {312, 1}, {486, 1}, {251, 1},
{377, 1}, {273, 1}, {276, 1}, {482, 1}, {394, 1}, {571, 1}, {391, 1}, {522, 1}}

```

```
Out[ ]= 1
```

```
Out[ ]= 752
```

Определить количество n во множестве, если при a81 экспериментах извлечения,

коллизия возникает с вероятностью a82. Ответ округлить до ближайшего большего целого.

```
In[ ]:= a81 = 60
a82 = 0.4
Solve[1 - Exp[(-a81 * (a81 - 1)) / (2 * N2)] == a82, N2]
```

решить у... показательная функция

```
Out[ ]:= 60
```

```
Out[ ]:= 0.4
```

Solve: Inverse functions are being used by Solve, so some solutions may not be found; use Reduce for complete solution information.

```
Out[ ]:= {{N2 -> 3464.98}}
```

```
In[ ]:= Ceiling[3464.98]
```

округление вверх

```
Out[ ]:= 3465
```

Найти значение функции Эйлера для числа x , которое определяется из соотношения $a^x + b = c \pmod n$, где $a = 47019$, $b = 47202$, $c = 19976$, $n = 13163$.

Ответ: ✓

```
In[ ]:= a1 = 47 019
        b1 = 47 202
        c1 = 19 976
        n1 = 13 163
        a11 = PowerMod[a1, -1, n1]
             |степень по модулю
```

Out[]:= 47 019

Out[]:= 47 202

Out[]:= 19 976

Out[]:= 13 163

Out[]:= 9305

```
In[ ]:= EulerPhi[Mod[c1 * a11 - b1 * a11, n1]]
             |функция... |остаток от деления
```

Out[]:= 10 330

Установить генератор случайных чисел в начальное состояние с параметром равным $2^{15} \pmod{157}$. Получить список из 10000 случайных простых чисел в диапазоне от 22000 до 41500. Найти произведение двух простых чисел, которые встречаются в списке с максимальной (применять функцию Max[]) и минимальной (применять функцию Min[]) частотой. В случае наличия чисел с одинаковыми частотами выбирать первое в списке.

Ответ: ✓

```
In[ ]:= SeedRandom[PowerMod[2, 15, 157]]
             |инициализа... |степень по модулю
        a2 = 22 000
        b2 = 41 500
        n2 = 10 000
        list2 = RandomPrime[{a2, b2}, n2]
                |случайное простое число
```

Out[]:= 22 000

Out[]:= 41 500

Out[]:= 10 000

```
In[ ]:= ans2max = Tally[list2][[
             |подсчитать
             FirstPosition[Tally[list2][[All, 2]], Max[Tally[list2][[All, 2]]][[1]], 1]]
             |позиция первого... |подсчитать |всё |ма... |подсчитать |всё
```

Out[]:= 39 317

```
In[ ]:= ans2min = Tally[list2][[
             |подсчитать
             FirstPosition[Tally[list2][[All, 2]], Min[Tally[list2][[All, 2]]][[1]], 1]]
             |позиция первого... |подсчитать |всё |м... |подсчитать |всё
```

Out[]:= 40 493

In[]:= **ans2max * ans2min**

Out[]:= 1 592 063 281

Определить энтропию сектора с номером 1109 виртуального флоппи-диска `prtest.flp` с точностью 5 знаков после запятой. Для округления результата применить функцию `N[]`. Пример ввода 5.55555

Ответ: 7.57882

псевдослучайных чисел для режима стирания по алгоритму Gutmann. Для ввода данных использовать предварительное преобразование с помощью утилиты `Converter.exe` и последующей функции `ReadList["file.dat", Number]`.

list3 = ReadList["D:\\SAUTOV'S PRIDE\\Рабочий стол\\kr3_.dat", Number]

| считать в ... | дифференцировать

| число

freq3 = Tally[list3]

| подсчитать

p3 = N[freq3[[All, 2]] / Length[list3]]

| численно ... | всё

| длина

np3 = Length[p3]

| длина

summp3 = Sum[p3[[i]], {i, np3}]

| сумма

ent3 = -Sum[p3[[i]] * Log[2, p3[[i]]], {i, np3}]

| сумма

| натуральный логарифм

Архив текстового файла `всптпне-164.zip` защищен паролем из 4-х символов, содержащих строчные и заглавные латинские буквы, а также все цифры. Один из символов пароля можно определить из следующего условия: сумма кода символа и кода позиции символа в пароле равна 58.5, а полуразность кода символа и кода позиции символа в пароле равна 8.5. Исключить пробелы и подсчитать число символов в тексте.

Ответ: 462

In[]:= **Solve[(x + y) / 2 == 58.5 && (x - y) / 2 == 8.5, {x, y}]**

| решить уравнения

Out[]:= { {x → 67., y → 50.} }

In[]:= **FromCharacterCode[50]**

| символ по его коду

Out[]:= 2

In[]:= **FromCharacterCode[67]**

| символ по его коду

Out[]:= C

In[]:= **text5 = "12345678980 0987654321"**

lengthtext5 = StringLength[text5] - StringCount[text5, " "]

| длина строки

| число случаев по образцу в стр.

Out[]:= 12345678980 0987654321

Out[]:= 21

Определить ожидаемое время раскрытия пароля длиной 7 символов и содержащего следующие наборы: (прописные русские, строчные латинские, прописные латинские), если скорость перебора пароля (пароль в секунду) равна обратному элементу числа 1587 по модулю 769. Ответ вводить как целое число суток.

Ответ: 302247

```
In[ ]:= a6 = 1587
         n6 = 769
         length6 = 7
         speed6 = PowerMod[a6, -1, n6]
               |_степень по модулю
```

```
Out[ ]:= 1587
```

```
Out[ ]:= 769
```

```
Out[ ]:= 7
```

```
Out[ ]:= 565
```

```
In[ ]:= alphabet = 32 + 26 + 26
```

```
Out[ ]:= 84
```

```
In[ ]:= varPass = alphabet^length6
```

```
Out[ ]:= 29 509 034 655 744
```

```
In[ ]:= Floor[varPass / (speed6 * 2 * 60 * 60 * 24)]
       |_округление вниз
```

```
Out[ ]:= 302 247
```

1. Определите число чисел взаимно простых с 7184 и меньших 7184. (3 584)

$a = 7184; k = 0$

$Do[If[GCD[a, i] == 1, k ++], \{i, a - 1, 0, - 1\}]; k$

2. Получить список из 2209 простых чисел. Сформировать из списка квадратную матрицу. Провести последовательно следующие операции: поменять местами строки 25 и 36; циклически сдвинуть 13-ю строку на 8 позиций влево. Поменять местами столбцы 34 и 20; циклически сдвинуть 29-й столбец на 37 позиций вверх. Рассчитать сумму элементов главной диагонали. (424 522)

$list = Table[Prime[i], \{i, 1, 2209\}]$

$list = Partition[list, Sqrt[2209]]$

$r36 = list[[36]]; r25 = list[[25]]$

$list[[36]] = r25; list[[25]] = r36$

$list[[13, All]] = RotateLeft[list[[13, All]], 8]$

$r20 = list[[All, 20]]; r34 = list[[All, 34]]$

$list[[20]] = r34; list[[34]] = r20$

$list[[All, 29]] = RotateLeft[list[[All, 29]], 37]$

$sum = 0; For[i = 1, i < Length[list] + 1, i ++, sum = sum + list[[i, i]]; sum$

Для нечетных чисел: $list = Table[2 * i - 1, \{i, 1, 2209\}]$

3. Сформировать множество целых чисел из интервала [40, 109] и множество простых чисел из интервала [76, 122]. Найти произведение элементов, принадлежащих обоим множествам. Если множества не пересекаются, ввести ответ 0. (6 868 087 171 373 809)

$spisok1 = Range[40, 109]$

$spisok2 = \{ \}$

$For[i = 76, i < NextPrime[122, - 1], i = NextPrime[i], spisok2 = Join[spisok2, \{NextPrime[i]\}]]$

$inter = Intersection[spisok1, spisok2]$

$roduct[inter[[i]], \{i, 1, Length[inter]\}]$

4. Дано число 3BA78BF(16). Перевести в 10 и посчитать разряды.

$b = 62552255$

$Length[IntegerDigits[b]]$

5. Дано число 7706, найти два ближайших простых числа к нему и посчитать их сумму по модулю 131.

$a1 = NextPrime[7706, 1]$

$a2 = NextPrime[7706, - 1]$

$Mod[(a1 + a2), 131]$

6. Составить список из заглавных букв английского алфавита, посчитать среднее значение кодов. (77.5)

$alpha = CharacterRange["A", "Z"]$

$codes = ToCharacterCode[StringJoin[alpha]]$

$summa = Sum[codes[[i]], \{i, 1, Length[codes]\}]$

$$N\left[\frac{summa}{Length[codes]}\right]$$

7. Сколько простых чисел содержится в диапазоне 83 – 536

$$PrimePi[536] - PrimePi[83]$$

Составить список из заглавных букв английского алфавита (в кодировке Unicode), посчитать среднее значение кодов.

```
In[2]:= AlphabetChar = CharacterRange["A", "Z"]
CharCode = ToCharacterCode[StringJoin[AlphabetChar]]
CharSum = Sum[CharCode[[i]], {i, 1, Length[CharCode]}]
CharAverage = N[CharSum / Length[CharCode]]

Out[2]= {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}

Out[3]= {65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90}

Out[4]= 2015

Out[5]= 77.5
```

Дано число 358138 [number], найти его обратное число [res1] в конечном поле GF(6502301) [GF(div)].

```
In[7]:= number = 358138
div = 6502301
res1 = PowerMod[number, -1, div]

Out[7]= 358138

Out[8]= 6502301

Out[9]= 4060720
```

Дано число 7706 [np], найти два ближайших к нему простых числа (b1, b2). Посчитать их сумму по модулю 131 [m].

```
In[11]:= np = 7706
m = 131
b1 = NextPrime[np, -1]
b2 = NextPrime[np, +1]
SumPrime = Mod[(b1 + b2), m]

Out[11]= 7706

Out[12]= 131

Out[13]= 7703

Out[14]= 7717

Out[15]= 93
```

Найти количество разрядов в двоичной записи шестнадцатиричного числа 33 A855.

```
In[17]:= Des = 16^^33A855
LenDv = IntegerLength[Des, 2]

Out[17]= 3385429

Out[18]= 22
```


Найти количество простых чисел в диапазоне от 17 [R1] до 29 [R2].

```
In[20]:= R1 = 17;
R2 = 29;
ListR = Table[s, {s, R1, R2}]
If[PrimeQ[R1], BeforeR1 = PrimePi[R1] - 1, BeforeR1 = PrimePi[R1]];
BeforeR2 = PrimePi[R2];
CountPrime = BeforeR2 - BeforeR1

Out[22]= {17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29}

Out[25]= 4
```

Получить множество целых чисел от [min1] до [max1] и множество простых чисел от [min2] до [max2].
Найти произведение элементов пересечения этих множеств.

```
In[27]:= min1 = 65;
max1 = 105;
List1 = Range[min1, max1]

Out[29]= {65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105}

In[30]:= min2 = 80;
max2 = 145;
If[PrimeQ[min2], t1 = PrimePi[min2], t1 = PrimePi[min2] + 1];
t2 = PrimePi[max2];
List2 = Table[Prime[x], {x, t1, t2}]

Out[34]= {83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139}

In[35]:= ListM = Intersection[List1, List2]
LenListM = Length[ListM]

Out[35]= {83, 89, 97, 101, 103}

Out[36]= 5

In[37]:= mult = 1; For[j = 1, j ≤ LenListM, j++, mult *= ListM[[j]]];
Print[mult]

7454155217
```

Составить список из нечетных чисел в диапазоне от nmin до nmax.

Сделать квадратную матрицу. Поменять местами строки str1 и str2.

Циклично сдвинуть строку str3 вправо (влево) на len1.

Поменять местами столбцы col1 и col2.

Циклично сдвинуть столбец col3 вверх (вниз) на len2.

Посчитать сумму элементов главной диагонали.

```
In[39]:= nmin = 1;
         nmax = 32;
         ListB = Table[2 * b - 1, {b, Ceiling[nmin/2], Floor[nmax/2]}]

Out[41]= {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31}

In[42]:= SqrtB = Sqrt[Length[ListB]]
         MatrixB = Partition[ListB, SqrtB]
         Dimensions[MatrixB, SqrtB]
         MatrixB // MatrixForm

Out[42]= 4

Out[43]= {{1, 3, 5, 7}, {9, 11, 13, 15}, {17, 19, 21, 23}, {25, 27, 29, 31}}

Out[44]= {4, 4}

Out[45]//MatrixForm=

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \\ 25 & 27 & 29 & 31 \end{pmatrix}$$


In[46]:= str1 = 2;
         str2 = 4;
         MatrixC = MatrixB;
         MatrixC[[str1]] = MatrixB[[str2]];
         MatrixC[[str2]] = MatrixB[[str1]];
         MatrixB = MatrixC;
         MatrixB // MatrixForm

Out[52]//MatrixForm=

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 25 & 27 & 29 & 31 \\ 17 & 19 & 21 & 23 \\ 9 & 11 & 13 & 15 \end{pmatrix}$$


In[53]:= str3 = 3;
         len1 = 1;
         MatrixB[[str3]] = RotateRight[MatrixB[[str3]], len1];
         MatrixB // MatrixForm

Out[56]//MatrixForm=

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 25 & 27 & 29 & 31 \\ 23 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 \end{pmatrix}$$

```

```

In[57]:= col1 = 1;
col2 = 3;
MatrixD1 = Transpose[MatrixB];
MatrixD2 = MatrixD1;
MatrixD2[[col1]] = MatrixD1[[col2]];
MatrixD2[[col2]] = MatrixD1[[col1]];
MatrixB = Transpose[MatrixD2];
MatrixB // MatrixForm

```

Out[64]//MatrixForm=

$$\begin{pmatrix} 5 & 3 & 1 & 7 \\ 29 & 27 & 25 & 31 \\ 19 & 17 & 23 & 21 \\ 13 & 11 & 9 & 15 \end{pmatrix}$$

```

In[65]:= col3 = 2;
len2 = 3;
MatrixF = Transpose[MatrixB];
MatrixF[[col3]] = RotateLeft[MatrixF[[col3]], len2];
MatrixB = Transpose[MatrixF];
MatrixB // MatrixForm

```

Out[70]//MatrixForm=

$$\begin{pmatrix} 5 & 11 & 1 & 7 \\ 29 & 3 & 25 & 31 \\ 19 & 27 & 23 & 21 \\ 13 & 17 & 9 & 15 \end{pmatrix}$$

```

In[71]:= ListD = Table[1, {g, 1, SqrtB}]
MultD = 1; For[u = 1, u ≤ SqrtB, u++, ListD[[u]] = MatrixB[[u, u]];
MultD *= ListD[[u]]]; ListD
Print[MultD]

```

Out[71]= {1, 1, 1, 1}

Out[72]= {5, 3, 23, 15}

5175

Справка. На виртуальной машине Examen #1 зайти в мой компьютер – сетевое окружение, набрать адрес сервера (на доске написан), открыть папку MSZI – TF..., откуда брать тексты и архивы. Флоппи-диск на виртуальную машину не устанавливать.

Установить генератор случайных чисел в начальное состояние с параметром (par) $2^{15}(\bmod 311)$. Получить список из 10 случайных простых чисел в диапазоне от 46 до 77. Найти произведение двух простых чисел, встречающихся в списке с максимальной (maxfr) и минимальной (minfr) частотами. В случае наличия чисел с одинаковыми частотами, выбирать первые в списке.

```
In[1]:= par = PowerMod[2, 15, 311]
SeedRandom[par]

Out[1]= 113

In[3]:= list1 = RandomPrime[{46, 77}, 10]
Out[3]= {73, 67, 73, 61, 47, 47, 61, 59, 61, 73}

In[4]:= listFREQ = Tally[list1]
Out[4]= {{73, 3}, {67, 1}, {61, 3}, {47, 2}, {59, 1}}

In[5]:= listSORT = Sort[listFREQ[[All, 2]], Greater]
Out[5]= {3, 3, 2, 1, 1}

In[6]:= i = 1; While[listFREQ[[i, 2]] != listSORT[[1]], i++];
maxfr = listFREQ[[i, 1]]
Out[7]= 73

In[8]:= u = 1; While[listFREQ[[u, 2]] != listSORT[[Length[listSORT]]], u++];
minfr = listFREQ[[u, 1]]
Out[9]= 67

In[10]:= res1 = minfr * maxfr
Out[10]= 4891
```

Найти значение функции Эйлера для числа x, которое определяется из соотношения: $a \cdot x + b = c \pmod n$, где $a = 34\,535$, $b = 34\,745$, $c = 26\,341$, $n = 11\,047$.

```
In[1]:= a = 34 535;
b = 34 745;
c = 26 341;
n = 11 047;

In[5]:= i = 1; While[Mod[Mod[a, n] * i, n] != Mod[c - b, n], i++];
Mod[Mod[a * i + b, n], n]
Mod[c, n]
EulerPhi[i]

Out[6]= 4247

Out[7]= 4247

Out[8]= 460
```

Определить ожидаемое время раскрытия пароля (tS) длиной (S) 7 символов и содержащего следующие наборы: {цифры, строчные русские, строчные латинские, прописные латинские}, если скорость перебора пароля (в символах в секунду) (R) равна обратному элементу числа 1939 по модулю 661. Ответ вводить как целое число суток.

```
In[1]:= S = 7
R = PowerMod[1939, -1, 661]

Out[1]= 7

Out[2]= 15

In[3]:= list1 = CharacterRange["0", "9"];
list2 = CharacterRange["а", "я"];
list3 = CharacterRange["a", "z"];
list4 = CharacterRange["A", "Z"];
list = Union[list1, list2, list3, list4]

Out[7]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н, о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я,
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, а, А, б, В, с, С, d, D, e, E, f, F, g, G, h, H, i, I, j, J, k,
K, l, L, m, M, n, N, o, O, p, P, q, Q, r, R, s, S, t, T, u, U, v, V, w, W, x, X, y, Y, z, Z}

In[8]:= A = Length[list]

Out[8]= 94

In[9]:= tS = IntegerPart[(1/2) * (A^S * 1/R) / 60 / 60 / 24]

Out[9]= 25 018 425
```

Архив текстового файла archive-118.zip защищен паролем из 4 символов, содержащих строчные и заглавные латинские буквы, а также все цифры. Один из символов пароля можно определить из следующего условия: полусумма кода символа и кода позиции символа в пароле равна 69, полуразность кода символа и кода позиции символа равна 20. Исключить пробелы и подсчитать число символов в тексте.

$$x = 20 + 69$$

$$y = 69 * 2 - x$$

Алгоритм действий:

- Решить систему линейных уравнений:
 x – код символа, y – код позиции символа
 $(x + y) / 2 = 69$
 $(x - y) / 2 = 20$
 $x = 69 + 20 = 89$
 $y = 2 * 69 - x = 49$

```
Out[21]= 89
Out[22]= 49

In[54]:= FromCharacterCode[x]
FromCharacterCode[y]

Out[54]= Y
Out[55]= 1
```

- Запустить установленную программу Advanced RAR Password Recovery, ввести параметры (набор – латинские строчные, латинские заглавные, все цифры) и установить маску (Y???). Открыть архив archive-118.zip, подобрать пароль.

- Открыть текстовый файл из архива в Word. Скопировать текст в Mathematica, удалить пробелы, если необходимо.

```
In[1]:= s1 = "Я памятник себе воздвиг нерукотворный!"
StringLength[s1]
s2 = StringReplace[s1, {" " -> ""}]
StringLength[s2]

Out[1]= Я памятник себе воздвиг нерукотворный!

Out[2]= 38

Out[3]= Япамятниксебезовдвигнерукотворный!

Out[4]= 34
```

Ответ: 34

Скачайте с сетевого диска (ftp-сервера) файл text-02.txt, расположенный в папке Texts и определите энтропию сообщения, содержащегося в нем. Ответ представить в битах, с 5 знаками после запятой.

Так: (по идее в байтах)

```
In[32]:= textfile = ReadList["C:\\MSZI\\text-02.txt", Byte]
        nByte = N[Entropy[2, textfile], 6]

Out[32]:= {204, 232, 248, 243, 240, 224, 44, 32, 236, 232, 248, 243, 240, 224, 44, 32, 226, 238,
          242, 32, 246, 226, 229, 242, 237, 224, 255, 32, 236, 232, 248, 243, 240, 224, 33}

Out[33]:= 3.90015
```

Или так: (по идее в битах)

```
In[29]:= textfile = ReadList["C:\\MSZI\\text-02.txt", Byte]
        nByte = N[Entropy[2, textfile], 10]
        nBit = N[nByte * 8, 7]

Out[29]:= {204, 232, 248, 243, 240, 224, 44, 32, 236, 232, 248, 243, 240, 224, 44, 32, 226, 238,
          242, 32, 246, 226, 229, 242, 237, 224, 255, 32, 236, 232, 248, 243, 240, 224, 33}

Out[30]:= 3.900153017

Out[31]:= 31.20122
```

Определите энтропию сектора с номером 795 виртуального флоппи-диска flptest.flp с точностью 5 знаков после запятой. Для округления результата применять функцию N[.]. Пример ввода: 5.55555. Уточнить у препода, что можно использовать в дробных выражениях, точку (.) или запятую (,)!

Алгоритм действий:

1. Вычислить 795 (номер вашего сектора) * 512 (количество байт в секторе). Пусть это X .
2. Открыть WinHEX, в меню Tools – Open disk выбрать флоппи-диск. После открытия выбрать в меню Position – Goto section (как-то так называется) и ввести значение X .
3. Конвертировать блок (convert) в файл с расширением .dat.
4. Найти энтропию файла:

```
In[36]:= myfile = ReadList["C:\\MSZI\\myfile.dat", Byte]
        nByte = N[Entropy[2, myfile], 6]

Out[36]:= {204, 232, 248, 243, 240, 224, 44, 32, 236, 232, 248, 243, 240, 224, 44, 32, 226, 238,
          242, 32, 246, 226, 229, 242, 237, 224, 255, 32, 236, 232, 248, 243, 240, 224, 33}

Out[37]:= 3.90015
```

1. Составить список из заглавных букв английского алфавита, посчитать среднее значение кодов.
`alphaChar = CharacterRange["A", "Z"]
codes = ToCharacterCode[StringJoin[alphaChar]]
sum = Sum[codes[[i]] , {i, 1, Length[codes]}]
N[sum / Length[codes]]`
2. Есть число $a=358138$, найти его обратное число в конечном поле GF (6502301)
`ExtendedGCD[358 138, 6 502 301]`
3. Дано число 7706, найти два ближайших простых числа к нему и посчитать их сумму по модулю 131.
`a3 = NextPrime[7706, 1]
a4 = NextPrime[7706, -1]
sum = Mod[(a3 + a4), 131]`
4. Дано число 3BA78BF(16). Перевести в 10 и посчитать разряды.
`a = 16^^3BA78BF
Length[IntegerDigits[a]]`
5. Составить список из нечетных чисел из диапазона 1 – 2209. Сделать квадратную матрицу. Поменять строки 35 и 2, сдвинуть циклично строку 6 вправо на 99, поменять местами столбцы 35 – 47. Циклично сдвинуть столбец 32 вверх на 99. Посчитать сумму элементов главной диагонали.
`spisok = Table[2* i - 1, {i, 2209}]
Sqrt[2209]; spisokQV = Partition[spisok, 47]
spisokQV // MatrixForm
Dimensions[spisokQV]
spisoktemp = spisokQV
spisoktemp[[35]] = spisokQV[[2]]
spisoktemp[[2]] = spisokQV[[35]]
spisoktemp
spisoktemp[[6]] = RotateRight[spisoktemp[[6]], 99]
spisoktemp1 = Transpose[spisoktemp]
Dimensions[spisoktemp1]
spisok111 = spisoktemp1
spisok111[[35]] = spisoktemp1[[47]]
spisok111[[47]] = spisoktemp1[[35]]
Transpose[spisok111] // MatrixForm`
6. Составить список целых чисел в диапазоне 65 – 105, список простых чисел из диапазона 80 – 145. Найти произведение пересечения. Если пересечения нет, то 0.
`spisok1 = Range[65, 105]
spisok21 = { }
For[i=80, i<NextPrime[145, -1], i = NextPrime[i], spisok21 = Join[spisok21, {NextPrime[i]}]]
spisok21
inter = Intersection[spisok1, spisok21]
Product[inter[[i]], { i, 1, Length[inter] }]`
7. Сколько простых чисел содержится в диапазоне 83 – 536
`PrimePi[536] – PrimePi[83]`

1. Определите число чисел взаимно простых с 7184 и меньших 7184. (3 584)

$a = 7184; k = 0$

$\text{Do}[\text{If}[\text{GCD}[a, i] == 1, k + +], \{i, a - 1, 0, -1\}]; k$

2. Получить список из 2209 простых чисел. Сформировать из списка квадратную матрицу. Провести последовательно следующие операции: поменять местами строки 25 и 36; циклически сдвинуть 13-ю строку на 8 позиций влево. Поменять местами столбцы 34 и 20; циклически сдвинуть 29-й столбец на 37 позиций вверх. Рассчитать сумму элементов главной диагонали. (424 522)

$\text{list} = \text{Table}[\text{Prime}[i], \{i, 1, 2209\}]$

$\text{list} = \text{Partition}[\text{list}, \text{Sqrt}[2209]]$

$\text{r36} = \text{list}[[36]]; \text{r25} = \text{list}[[25]]$

$\text{list}[[36]] = \text{r25}; \text{list}[[25]] = \text{r36}$

$\text{list}[[13, \text{All}]] = \text{RotateLeft}[\text{list}[[13, \text{All}]], 8]$

$\text{r20} = \text{list}[[\text{All}, 20]]; \text{r34} = \text{list}[[\text{All}, 34]]$

$\text{list}[[20]] = \text{r34}; \text{list}[[34]] = \text{r20}$

$\text{list}[[\text{All}, 29]] = \text{RotateLeft}[\text{list}[[\text{All}, 29]], 37]$

$\text{sum} = 0; \text{For}[i = 1, i < \text{Length}[\text{list}] + 1, i + +, \text{sum} = \text{sum} + \text{list}[[i, i]]; \text{sum}$

Для нечетных чисел: $\text{list} = \text{Table}[2 * i - 1, \{i, 1, 2209\}]$

3. Сформировать множество целых чисел из интервала [40, 109] и множество простых чисел из интервала [76, 122]. Найти произведение элементов, принадлежащих обоим множествам. Если множества не пересекаются, ввести ответ 0. (6 868 087 171 373 809)

$\text{spisok1} = \text{Range}[40, 109]$

$\text{spisok2} = \{ \}$

$\text{For}[i = 76, i < \text{NextPrime}[122, -1], i = \text{NextPrime}[i], \text{spisok2} = \text{Join}[\text{spisok2}, \{\text{NextPrime}[i]\}]]$

$\text{inter} = \text{Intersection}[\text{spisok1}, \text{spisok2}]$

$\text{roduct}[\text{inter}[[i]], \{i, 1, \text{Length}[\text{inter}]\}]$

4. Дано число 3BA78BF(16). Перевести в 10 и посчитать разряды.

$b = 62552255$

$\text{Length}[\text{IntegerDigits}[b]]$

5. Дано число 7706, найти два ближайших простых числа к нему и посчитать их сумму по модулю 131.

$a1 = \text{NextPrime}[7706, 1]$

$a2 = \text{NextPrime}[7706, -1]$

$\text{Mod}[(a1 + a2), 131]$

6. Составить список из заглавных букв английского алфавита, посчитать среднее значение кодов. (77.5)

$\text{alpha} = \text{CharacterRange}["A", "Z"]$

$\text{codes} = \text{ToCharacterCode}[\text{StringJoin}[\text{alpha}]]$

$\text{summa} = \text{Sum}[\text{codes}[[i]], \{i, 1, \text{Length}[\text{codes}]\}]$

$N[\text{summa}/\text{Length}[\text{codes}]]$

7. Сколько простых чисел содержится в диапазоне 83 – 536

$\text{PrimePi}[536] - \text{PrimePi}[83]$

Функции Эйлера:

```
In[37]:= x = (Mod[c, n] - b) * PowerMod[a, -1, n]
Out[37]= -427 649 556

In[38]:= EulerPhi[Mod[x, n]]
Out[38]= 2244
```

Энтропия файла:

```
In[90]:= plntext = ReadList["C:\Text-39.txt", Byte];
In[91]:= N[Entropy[2, plntext], 6]
Out[91]= 4.42733
```

Энтропия диска:

Скорость:

```
In[30]:= S = 5; R = PowerMod[1148, -1, 547]
Out[30]= 233

In[31]:= T = IntegerPart[(1/2 * 10^5 * 1/R) / 60 / 60 / 24]
Out[31]= 0
```

Частота:

```
In[1]:= SeedRandom[Mod[2^15, 313]]
pr = RandomPrime[{47 000, 79 000}, 10 000];

In[3]:= prr = Tally[pr];
SortBy[Tally[pr], Last];

In[9]:= i = 1; While[prr[[i, 2]] ≠ 1, i++];
min = prr[[i, 1]]
j = 1; While[prr[[j, 2]] ≠ 12, j++];
max = prr[[j, 1]]
proiz = max * min
```

Пароль от архива:

```
x = 20 + 69
y = 69 * 2 - x

Out[21]= 89
Out[22]= 49

In[54]:= FromCharacterCode[x]
FromCharacterCode[y]

In[35]:= StringJoin[StringCases[stroka, CharacterRange["a", "я"]]]
StringLength[stroka]

Out[35]= отецпошлидомойяникакнемомуломатьлюсиндучтобыяшелдомойиз:
тынебудуноонавсеравночточленсемьипричиталамиссистисне:

Out[54]= Y
Out[55]= 1

Out[36]= 371
```

НИУ МЭИ

Лабораторная работа №1

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

Преподаватель: Рытов А.А.

Москва 2016г.

1. Создать три числа, являющиеся отображением Вашей фамилии, имени и отчества, с использованием соответствия между русским алфавитом и множеством целых

$$\bar{Z}_{32} = \{1, 2, 3, \dots, 32\}.$$

| Буква | Число | Буква | Число | Буква | Число | Буква | Число |
|-------|-------|-------|-------|-------|-------|-------|-------|
| а | 1 | И | 9 | р | 17 | ш | 25 |
| б | 2 | Й | 10 | с | 18 | щ | 26 |
| в | 3 | К | 11 | т | 19 | ь | 27 |

| | | | | | | | |
|---|---|---|----|---|----|---|----|
| Г | 4 | Л | 12 | у | 20 | Ы | 28 |
| Д | 5 | М | 13 | Ф | 21 | Ъ | 29 |
| Е | 6 | Н | 14 | Х | 22 | Э | 30 |
| Ж | 7 | О | 15 | Ц | 23 | Ю | 31 |
| З | 8 | П | 16 | Ч | 24 | Я | 32 |

```
In[2]:= imya = 3 112 932
        otchestvo = 634 614 273 141
        familiya = 32 112 025 614 111 531
```

```
Out[2]= 3 112 932
```

```
Out[3]= 634 614 273 141
```

```
Out[4]= 32 112 025 614 111 531
```

2. Перевести три числа в двоичную, восьмеричную и шестнадцатеричную формы. Использовать функцию **BaseForm[expr,n]**- возвращает выражение expr в форме числа с основанием n, которое указывается как подстрочный индекс.

```
In[69]:= BaseForm[imya, 2]
        BaseForm[imya, 8]
        BaseForm[imya, 16]

Out[69]/BaseForm=
10111101111111111001002

Out[70]/BaseForm=
136777448

Out[71]/BaseForm=
2f7fe416

In[11]:= BaseForm[familiya, 2]
        BaseForm[familiya, 8]
        BaseForm[familiya, 16]

Out[11]/BaseForm=
11100100001010110110111100101111011111101010011001010112

Out[12]/BaseForm=
16205333627576514538

Out[13]/BaseForm=
7215b797bf532b16
```

3. Восстановить числа в десятичной форме, используя сначала конструкцию **base^^digits** (основание^^число).

```

In[15]:= 8^^13677744
         16^^2f7fe4
         2^^10010011111000001111101010101010001110101
         8^^11170175252165
         16^^93c1f55475

Out[15]= 3 112 932

Out[16]= 3 112 932

Out[17]= 634 614 273 141

Out[18]= 634 614 273 141

Out[19]= 634 614 273 141

```

4. Получить списки цифр (символов), составляющих числа в десятичной, двоичной, и шестнадцатеричной формах. Использовать функцию IntegerDigits[n,b]: n- число, b- основание.

```

In[20]:= IntegerDigits[3112932, 10]
Out[20]= {3, 1, 1, 2, 9, 3, 2}

In[21]:= IntegerDigits[1011110111111111100100, 2]
Out[21]= {1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1,
          1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0}

In[22]:= IntegerDigits[3112932, 2]
Out[22]= {1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0}

In[23]:= IntegerDigits[3112932, 16]
Out[23]= {2, 15, 7, 15, 14, 4}

```

5. Провести операцию деления "числа-фамилии" на "число-имя". Результат целочисленного деления перевести в вещественную форму с помощью функции N[expr].

```

In[25]:= N[familiya / imya]
Out[25]= 1.03157 × 1010

In[26]:= ve = N[familiya / imya, 10]
Out[26]= 1.031568490 × 1010


In[27]:= 1.031568489581896777`10.*^10
Out[27]= 1.031568490 × 1010

```

6. Найти целую и дробную часть полученного в п.5 вещественного числа. Использовать соответственно функции IntegerPart[x], FractionalPart[x].

```
In[72]:= IntegerPart[ve]  
FractionalPart[ve]
```

```
Out[72]= 10 315 684 895
```

```
Out[73]= 
```

7. Провести приведение вещественного числа к ближайшим целым с помощью следующих функций: Floor[x]- возвращает наибольшее целое число, не превышающее данного x; Ceiling[x]- возвращает значение наименьшего целого числа, большего или равного x.

```
In[30]:= Floor[ve]
```

```
Out[30]= 10 315 684 895
```

```
In[31]:= Ceiling[ve]
```

```
Out[31]= 10 315 684 896
```

8. Определить значения максимально и минимально возможных значений чисел, с которыми оперирует система Mathematica 7. Использовать функции \$MaxMachineNumber и \$MinMachineNumber.

```
In[32]:= $MaxMachineNumber  
$MinMachineNumber
```

```
Out[32]=  $1.79769 \times 10^{308}$ 
```

```
Out[33]=  $2.22507 \times 10^{-308}$ 
```

9. Разбить "число-отчество" на три числа, состоящие из трех цифр (например: 166; 191; 715, остаток отбросить). Получить три простых числа, номер которых определяется полученными значениями.

```
In[34]:= Prime[634]  
Prime[614]  
Prime[273]
```

```
Out[34]= 4691
```

```
Out[35]= 4519
```

```
Out[36]= 1753
```

10. Найти простые числа с номерами 99;100;101.

Out[39]= 547

Out[42]= 547

Out[43]= 99

Out[48]= 971

179 769 323 486 231 570 814 327 423 731 704 556 798 070 867 826 844 096 598 917 476 035 187 260 780 028 538 760 598 888 832 766 878 171 540 485 953 514 382 444 234 321 524 828 464 182 768
447 946 703 537 516 986 049 910 576 951 282 076 248 490 090 389 328 944 075 666 508 455 133 964 104 583 236 903 222 388 166 808 559 332 123 346 274 797 826 204 144 723 146 738 177 180
819 289 881 250 044 026 104 124 888 283

15

```
In[74]:= IntegerLength[maxPrime, 10]
IntegerLength[maxPrime, 2]
IntegerLength[maxPrime, 16]
```

```
Out[74]= 309
```

```
Out[75]= 1024
```

```
Out[76]= 256
```

16. Определить номер "maxPrime".

```
In[101]:= nom = Floor[RiemannR[maxPrime]]
```

```
Out[101]= 259 631 570 116 784 191 906 652 115 991 352 385 522 944 592 236 674 219 516 978 380 035 319 589 625 344 495 829 963 409 005 589 645 411 928 336 380 836 750 184 267 189 994 241 527 034 214 668 136 506 744 118 702 940 494 625 259 558 623 243 659 088 347 846 224 654 404 348 166 860 147 053 778 610 355 555 171 945 909 387 513 546 523 404 514 392 365 261 526 469 373 045 364 413 262 130 574 360 663 403 598 751
```

17. Получить три случайных целых числа в диапазоне от 0 до $imax = 255$, последовательно применяя функцию RandomInteger[imax].

```
In[55]:= RandomInteger[255, 3]
```

```
Out[55]= {139, 10, 154}
```

18. Установить генератор псевдослучайных чисел в начальное состояние, которое определяется "числом-фамилией". Использовать функцию SeedRandom[n]- переводит генератор псевдослучайных чисел в начальное состояние, определяемое параметром n.

```
In[56]:= SeedRandom[familiya]
```

19. Получить три случайных целых числа в диапазоне от 0 до $imax = 1000$.

```
In[57]:= RandomInteger[1000, 3]
```

```
Out[57]= {294, 949, 718}
```

20. Повторно получить последовательность из трех чисел п.19.

```
In[58]:= RandomInteger[1000, 3]
```

```
Out[58]= {186, 634, 185}
```

21. Найти случайное число, которое находится в диапазоне "число-имя" $\pm 10 \times N$, где N – номер по списку в группе. Использовать функцию RandomInteger[{imin,imax}].

```
In[59]:= RandomInteger[{imya, 10 * 13}]
```

```
Out[59]= 939 567
```

22. Сформировать последовательность из 40-N случайных чисел, находящихся в диапазоне от 0 до 128. Использовать функцию RandomInteger[range, n]: range - верхняя граница диапазона, n - число случайных чисел.

```
In[60]:= RandomInteger[128, 40 - 13]
```

```
Out[60]= {62, 53, 74, 14, 90, 72, 4, 126, 11, 2, 29, 104, 56, 81, 37, 9, 89, 57, 33, 119, 45, 111, 29, 21, 50, 96, 88}
```

23. Получить три простых случайных целых числа в диапазоне от 2 до $imax = 512$. Использовать функцию RandomPrime[imax].

```
In[61]:= RandomPrime[{2, 512}, 3]
```

```
Out[61]= {227, 13, 131}
```

24. Повторно получить последовательность из трех простых чисел п.23.

```
In[62]:= RandomPrime[{2, 512}, 3]
```

```
Out[62]:= {239, 103, 173}
```

25. Найти простое случайное число, которое находится в диапазоне "число-имя" $\pm 10 \times N$, где N – номер по списку в группе. Использовать функцию `RandomPrime[{imin,imax}]`.

```
In[102]:= RandomPrime[{inya - 10 * 13, inya + 10 * 13}]
```

```
Out[102]:= 3 112 807
```

26. Сформировать последовательность из 40-N простых случайных чисел, находящихся в диапазоне от 0 до 1024. Использовать функцию `RandomPrime [range, n]`

```
Out[94]:= RandomPrime[1024, 40 - 13]
```

```
Out[94]:= {569, 131, 727, 293, 821, 109, 641, 997, 1009, 719, 251, 139, 347, 311, 541, 19, 277, 101, 641, 587, 139, 853, 421, 743, 277, 571, 691}
```


НИУ МЭИ

Лабораторная работа №2

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

Преподаватель: Рытов А.А.

Москва 2016г.

1. Выбрать три имени-идентификатора для переменных, соответствующих "числу-фамилии", "числу-имени" и "числу-отчеству" (Лаб.1) в соответствии с принятыми в системе Mathematica правилами: sssss – имя объекта, заданного пользователем, имена переменных должны начинаться с буквы.

`imya otchestvo familiya`

2. Присвоить переменным числовые значения, используя операцию присваивания: var=value, где var – имя переменной, value – её значение. При таком определении переменные в системе Mathematica являются глобальными.

```
imya = 3 112 932
otchestvo = 634 614 273 141
familiya = 32 112 025 614 111 531

3 112 932

634 614 273 141

32 112 025 614 111 531
```

3. Ввести переменную для результата деления и, используя существующие переменные, провести операцию деления "числа-фамилии" на "число-имя". Результат целочисленного деления перевести в вещественную форму с помощью функции N[expr].

```
In[108]:= del = N[familiya/imya]

Out[108]:= 1.03157 × 1010
```

4. Выбрать три имени-идентификатора для символьных переменных, соответствующих фамилии, имени и отчеству, и присвоить им три символьные строки: фамилия, имя отчество. Символьные строки задаются цепочкой символов в кавычках, например "sssss".

```
In[147]:= imya1 = "Юлия"
otchestvo1 = "Евгеньевна"
familiya1 = "Якушенкова"

Out[147]:= Юлия

Out[148]:= Евгеньевна

Out[149]:= Якушенкова
```

5. Подготовить две переменные (например: list_family и list_familyN), и присвоить им значения списков, состоящих из символов фамилии и цифр, из "числа-фамилии". Списки (lists) являются наиболее общим видом сложных (множественных) данных в системе Mathematica. Они представляют совокупность однотипных и разнотипных данных, сгруппированных с помощью фигурных скобок. Например: {1,2,3} – список из трех целых чисел; {a,b,v} – список из трех символьных данных. В дальнейшем предполагается, что все списки должны иметь оригинальные имена-идентификаторы.

```
In[110]:= listfamily = {"Я", "к", "у", "ш", "е", "н", "к", "о", "в", "а"}
listfamilyN = {32, 11, 20, 25, 6, 14, 11, 15, 3, 1}

Out[110]:= {Я, к, у, ш, е, н, к, о, в, а}

Out[111]:= {32, 11, 20, 25, 6, 14, 11, 15, 3, 1}
```

6. Выделить тремя способами второй и четвертый элементы из сформированных списков. Для выделения элементов списка list используются двойные квадратные скобки: list[[i]] – выделяет i-ый элемент списка с его начала (если i < 0 – с конца); list[[i,j, ...]] – выделяет i-ый, j-ый и т.д. элементы списка. Функция Part[list,i] – выделяет i-ый элемент списка list.

Out[118]= III

Out[151]= 10

Output: [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, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87]

```
Out[152]= {1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400}
```

```
Out[119]= {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576}
```

2007/02/26: Newham

```
Cv(12)BaseForm
```

[illegible]

14. Подготовить список (rand200) из 200 случайных целых чисел из диапазона 1÷100. Провести сортировку списка с помощью функции Sort[list], которая располагает элементы списка в каноническом порядке. Определить число элементов со значениями 10+N, 20+N, 30+N. Использовать функцию Count[list,pattern] – возвращает количество элементов в списке list, которые соответствуют образцу pattern.

15. Получить элементы списка `rand200` с номерами от $10+N$ до $30+N$. Использовать функцию `Take[list, {m,n}]` – возвращает элементы списка с порядковыми номерами от m до n .

16. Провести операцию "поворота" списка влево на $5+N$ позиций: функция `RotateLeft[list,n]`- циклический сдвиг списка влево на n позиций.

Провести операцию "поворота" списка вправо на $5+N$ позиций: функция `RotateRight[list,n]`.

17. Определить число копий, для каждого элемента ,содержащегося в списке rand200. Функция Tally[list].

18. Подготовить три списка (listrng15, listrng20, listrng25), ранжированных в интервалах $1 \div 15+N$; $5 \div 20+N$; $10 \div 25+N$, представляющих собой три конечных множества целых чисел.

```
In[131]:= listrng15 = Range[1, 15 + 13]
Out[131]:= {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}

In[132]:= listrng20 = Range[5, 20 + 13]
Out[132]:= {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}

In[133]:= listrng25 = Range[10, 25 + 13]
Out[133]:= {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}
```

19. Во множестве listrng15 найти элементы, которые не содержатся в множествах listrng20, listrng25. Использовать функцию Complement[list,list1,list2,...] – возвращает список list с элементами, которые не содержатся ни в одном из списков list1,list2,...

```
In[134]:= Complement[listrng15, listrng20, listrng25]
Out[134]:= {1, 2, 3, 4}
```

20. Определить пересечение множеств listrng15, listrng20, listrng25. Использовать функцию Intersection[list1,list2,...] – возвращает упорядоченный список элементов, общих для всех списков listi.

```
In[135]:= Intersection[listrng15, listrng20, listrng25]
Out[135]:= {10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28}
```

21. Провести операцию конкатенации для множеств listrng15, listrng20, listrng25. Использовать функцию Join[list1,list2,...] – объединяет множества (списки) в единую цепочку.

```
In[136]:= Join[listrng15, listrng20, listrng25]
Out[136]:= {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, 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, 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}
```

22. Провести операцию объединения множеств listrng15, listrng20, listrng25. Использовать функцию Union [list1,list2,...] – удаляет повторяющиеся элементы списков и возвращает отсортированный список всех различающихся между собой элементов, принадлежащих любому из данных списков listi.

```
In[137]:= Union[listrng15, listrng20, listrng25]
Out[137]:= {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}
```

23. Удалить повторяющиеся элементы из списка rand200.

```
In[138]:= Union[list]
Out[138]:= {1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 39, 40, 41, 42, 44, 45, 46, 47, 48, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 70, 72, 73, 74, 75, 76, 77, 78, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 96, 97, 98, 99, 100}
```

24. Определить количество чисел в диапазоне от 1 до 100, отсутствующих в списке rand200, найти эти числа и представить в виде списка.

```
In[139]:= sort = Complement[list, {Range[100]}]
Length[sort]
Out[139]:= 1
Out[140]:= {1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 39, 40, 41, 42, 44, 45, 46, 47, 48, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 70, 72, 73, 74, 75, 76, 77, 78, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 96, 97, 98, 99, 100}
```

НИУ МЭИ

Лабораторная работа №3

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

Преподаватель: Рытов А.А.

1. Создать пустую строку с именем `myStringN = ""` (N- номер по списку в группе) и скопировать в неё из файла `plaintext.doc` страницу с номером N. Do you want escapes inserted in the text you are pasting?
⇒ Yes.

```
In[201]:= myString13 =
"использованию карт, а также введение ограничений на количество
наличных денег, которые может получить клиент в течение
одного дня (для защиты от использования украденных карт) .
\тОднако этот режим возможен лишь при наличии надежных каналов
связи между банкоматами и банком, что делает его довольно
дорогим. Кроме того, наличие канала связи порождает и другие
угрозы безопасности по сравнению с автономным режимом работы.
Это – анализ трафика между банкоматом и главным компьютером
и имитация работы главного компьютера компьютером зло-умышленника.
При анализе трафика можно получить информацию о счетах,
суммах, условиях пла-тежей и т.п. При имитации работы главного
компьютера банка компьютер злоумышленника может выдавать
положительный ответ на запрос банкомата о результатах
идентификации/аутентификации.
\тСети банкоматов являются в настоящее время распространенной
формой эксплуатации банкоматов, в которой участвуют несколько
банков [22, 123]. Банки-участники такой сети преследуют
следующие цели:
•\туменшение стоимости операций для участников;
•\тразделение затрат и риска при внедрении новых видов услуг
между участниками;
•\тпреодоление географических ограничений и соответственно
повышение субъективной ценности услуг для потребителей.
```

2. Определить полное число байтов, используемых для хранения символов в строке и число символов в строке :⇒ `StringByteCount[], StringLength[]`.

```
In[159]:= StringByteCount[myString13]
```

```
Out[159]:= 5682
```

```
In[160]:= StringLength[myString13]
```

```
Out[160]:= 3132
```

3. Получить список строчных букв русского алфавита, объединить их в строку и получить список кодов, соответствующих строчным буквам :⇒ `CharacterRange[], StringJoin[], Characters[], ToCharacterCode[]`.

```
In[161]:= alpha = CharacterRange["а", "я"]
```

```
Out[161]:= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н,
о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я}
```

```
In[202]:= str1 = StringJoin[alpha]
```

```
Out[202]:= абвгдежзийклмнопрстуфхцчшщъыьэюя
```

```

In[203]:= Characters[str1]
Out[203]:= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н,
           о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я}

In[204]:= ToCharacterCode[str1]
Out[204]:= {1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081,
           1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092,
           1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103}

```

4. Получить список прописных букв русского алфавита, объединить их в строку и получить список кодов, соответствующих прописным буквам.

```

In[205]:= Alpha = CharacterRange["А", "Я"]
          str2 = StringJoin[Alpha]
          Characters[str2]
          ToCharacterCode[str2]

Out[205]:= {А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н,
           О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я}

Out[206]:= АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ

Out[207]:= {А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н,
           О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я}

Out[208]:= {1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049,
           1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060,
           1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071}

```

5. Определить код символа "пробел". Определить коды символов ":" , "-" , ";".

```

In[169]:= escape = ToCharacterCode[" "]
          simbol1 = ToCharacterCode[":"]
          simbol2 = ToCharacterCode["-"]
          simbol3 = ToCharacterCode[";"]

Out[169]:= {32}

Out[170]:= {58}

Out[171]:= {45}

Out[172]:= {59}

```

6. Определить коды цифр.

```

In[173]:= digits = CharacterRange["0", "9"];
          cifr = StringJoin[digits]
          digcode = ToCharacterCode[cifr]

Out[174]:= 0123456789

Out[175]:= {48, 49, 50, 51, 52, 53, 54, 55, 56, 57}

```


7. Объединить списки прописных и строчных букв с помощью функции `Riffle[]`.

```
In[176]:= Riffle[Alpha, alpha]
```

```
Out[176]:= {A, a, Б, б, В, в, Г, г, Д, д, Е, е, Ж, ж, З, з, И, и, Й, й,  
            К, к, Л, л, М, м, Н, н, О, о, П, п, Р, р, С, с, Т, т, У, у, Ф, ф,  
            Х, х, Ц, ц, Ч, ч, Ш, ш, Щ, щ, Ъ, ъ, Ы, ы, Ь, ь, Э, э, Ю, ю, Я, я}
```

8. Провести замену прописных букв на строчные в строке `myStringN`, используя функцию `StringReplace["string",{ "s1"→"spl","s2"→"sp2",...}]`. Ввод управляющего символа подстановки `→` производится при наборе следующей последовательности символов: `Esc -> Esc`.

```
myString=StringReplace[myString13,Table[FromCharacterCode[1040+i]->FromCharacterCode[1072+i],{i,0,31}]]
```

```
In[179]:= myString = StringReplace[myString13,  
    {"A" → "a", "Б" → "б", "В" → "в", "Г" → "г", "Д" → "д", "Е" → "е",  
     "Ж" → "ж", "З" → "з", "И" → "и", "Й" → "й", "К" → "к", "Л" → "л",  
     "М" → "м", "Н" → "н", "О" → "о", "П" → "п", "Р" → "р", "С" → "с",  
     "Т" → "т", "У" → "у", "Ф" → "ф", "Х" → "х", "Ц" → "ц", "Ч" → "ч",  
     "Ш" → "ш", "Щ" → "щ", "Ъ" → "ъ", "Ы" → "ы", "Ь" → "ь", "Э" → "э",  
     "Ю" → "ю", "Я" → "я"}]
```

9. Определить число символов "." в строке: \Rightarrow `StringCount[]`.

```
In[180]:= StringCount[myString13, "."]
```

```
Out[180]:= 28
```

10. Определить позиции размещения "." в строке: \Rightarrow `StringPosition[]`

```
In[181]:= StringPosition[myString13, "."]
```

```
Out[181]:= {{176, 176}, {305, 305}, {419, 419}, {546, 546}, {634, 634},  
            {636, 636}, {806, 806}, {948, 948}, {1243, 1243}, {1419, 1419},  
            {1421, 1421}, {1423, 1423}, {1549, 1549}, {1627, 1627},  
            {1807, 1807}, {1810, 1810}, {1813, 1813}, {1906, 1906},  
            {1964, 1964}, {1967, 1967}, {2156, 2156}, {2159, 2159}, {2450, 2450},  
            {2453, 2453}, {2606, 2606}, {2609, 2609}, {2753, 2753}, {2978, 2978}}
```

11. Из строки, в которой была произведена замена прописных букв (см. п.8), исключить все символы (знаки препинания, цифры и т.д.), кроме строчных букв. Полученный список преобразовать в новую строку и определить ее длину: \Rightarrow `StringCases[]`.

```
In[182]:= str = StringCases[myString13, alpha]
```

```
In[183]:= strjoin = StringJoin[str]
```

```
In[184]:= StringLength[strjoin]
```

```
Out[184]:= 2497
```

12. Создать список (см. п.3) из 32 букв русского алфавита с помощью функции `CharacterRange[]`.

```
In[185]:= alpha
Out[185]= {а, б, в, г, д, е, ж, з, и, й, к, л, м, н,
           о, п, р, с, т, у, ф, х, ц, ч, ш, щ, ъ, ы, ь, э, ю, я}
```

13. Сформировать из полученного списка матрицу, состоящую из 4-х строк и 8-и столбцов. Применить функцию Partition[].

```
In[209]:= matr = Partition[alpha, 8]
          matr // TableForm

Out[209]= {{а, б, в, г, д, е, ж, з}, {и, й, к, л, м, н, о, п},
           {р, с, т, у, ф, х, ц, ч}, {ш, щ, ъ, ы, ь, э, ю, я}}

Out[210]//TableForm=
  а  б  в  г  д  е  ж  з
  и  й  к  л  м  н  о  п
  р  с  т  у  ф  х  ц  ч
  ш  щ  ъ  ы  ь  э  ю  я
```

14. Проверить размерность полученного списка:-> Dimensions[].

```
In[212]:= Dimensions[matr]
Out[212]= {4, 8}
```

15. Сформировать массив, который имеет 4×8 элементов с нулевыми начальными индексами (index origins):->Array[].

```
In[296]:= Array[matriza, {4, 8}, 0]

Out[296]= {{matriza[0, 0], matriza[0, 1], matriza[0, 2], matriza[0, 3], matriza[0, 4], matriza[0, 5], matriza[0, 6], matriza[0, 7]},
           {matriza[1, 0], matriza[1, 1], matriza[1, 2], matriza[1, 3], matriza[1, 4], matriza[1, 5], matriza[1, 6], matriza[1, 7]},
           {matriza[2, 0], matriza[2, 1], matriza[2, 2], matriza[2, 3], matriza[2, 4], matriza[2, 5], matriza[2, 6], matriza[2, 7]},
           {matriza[3, 0], matriza[3, 1], matriza[3, 2], matriza[3, 3], matriza[3, 4], matriza[3, 5], matriza[3, 6], matriza[3, 7]}}
```

16. Провести инициализацию элементов массива, присвоив каждому элементу массива соответствующее значение элемента матрицы, например a[0,0]=list[[1,1]] и т.д.

```
In[297]:= Do[matriza[i, j] = matr[[i + 1, j + 1]], {i, 0, 3}, {j, 0, 7}]
```

17. Вывести на экран элементы массива представленные в матричной форме Array[,,]//MatrixForm.

```
In[300]:= Array[matriza, {4, 8}, 0] // MatrixForm

Out[300]//MatrixForm=
  ( а  б  в  г  д  е  ж  з
    и  й  к  л  м  н  о  п
    р  с  т  у  ф  х  ц  ч
    ш  щ  ъ  ы  ь  э  ю  я )
```

18. Установить ГСЧ (генератор случайных чисел) в начальное состояние с параметром - буквой русского алфавита, соответствующей номеру по списку в группе

```
In[273]:= ToCharacterCode["м"]
Out[273]:= {1084}

In[274]:= SeedRandom[1084]
```

19. Провести операции случайной перестановки элементов для каждой из строк: \Rightarrow RandomSample[].

```
In[590]:= For[i = 1, i < 4, i++, arr[[i]] = RandomSample[arr[[i]]]]
arr // MatrixForm

Out[591]//MatrixForm=

$$\begin{pmatrix} \text{з} & \text{ж} & \text{а} & \text{д} & \text{е} & \text{г} & \text{б} & \text{в} \\ \text{н} & \text{к} & \text{и} & \text{м} & \text{й} & \text{п} & \text{л} & \text{о} \\ \text{с} & \text{р} & \text{ч} & \text{х} & \text{ц} & \text{ф} & \text{т} & \text{у} \\ \text{ы} & \text{э} & \text{щ} & \text{ъ} & \text{ь} & \text{ю} & \text{ш} & \text{я} \end{pmatrix}$$

```

20. Провести повторную инициализацию массива элементами случайной перестановки алфавита.

```
In[595]:= For[k = 1; i = 0, i < 4, i++, For[j = 0, j < 8, j++, arr[[i + 1, j + 1]] = alpha[[k]]; k++]]
arr // MatrixForm

Out[596]//MatrixForm=

$$\begin{pmatrix} \text{а} & \text{б} & \text{в} & \text{г} & \text{д} & \text{е} & \text{ж} & \text{з} \\ \text{и} & \text{й} & \text{к} & \text{л} & \text{м} & \text{н} & \text{о} & \text{п} \\ \text{р} & \text{с} & \text{т} & \text{у} & \text{ф} & \text{х} & \text{ц} & \text{ч} \\ \text{ш} & \text{щ} & \text{ъ} & \text{ы} & \text{ь} & \text{э} & \text{ю} & \text{я} \end{pmatrix}$$

```

21. Вывести список, состоящий из элементов строки массива с номером $N_{\text{mod}3}$ и столбца массива с номером $N_{\text{mod}7}$.

```
In[612]:= Join[arr[[Mod[13, 3], All]], arr[[All, Mod[13, 7]]]]
Out[612]:= {а, б, в, г, д, е, ж, з, е, н, х, э}

In[614]:= Mod[13, 3]
Mod[13, 7]

Out[614]:= 1

Out[615]:= 6
```

22. Вывести отображение массива в матричной форме: \Rightarrow Array[]/MatrixForm.

```
In[617]:= Array[matrixa, {4, 8}, 0] // MatrixForm

Out[617]//MatrixForm=

$$\begin{pmatrix} \text{а} & \text{б} & \text{в} & \text{г} & \text{д} & \text{е} & \text{ж} & \text{з} \\ \text{и} & \text{й} & \text{к} & \text{л} & \text{м} & \text{н} & \text{о} & \text{п} \\ \text{р} & \text{с} & \text{т} & \text{у} & \text{ф} & \text{х} & \text{ц} & \text{ч} \\ \text{ш} & \text{щ} & \text{ъ} & \text{ы} & \text{ь} & \text{э} & \text{ю} & \text{я} \end{pmatrix}$$

```

23. Найти элемент массива, расположенный в столбце на одну позицию ниже буквы, соответствующей номеру по списку в алфавите: $\Rightarrow \text{Array}[], \text{Position}[], \text{Part}[], \text{Mod}[]$.

```
In[818]:= For[i = 1, i ≤ 4, i++,  
            For[j = 1, j ≤ 8, j++,  
              If[arr[[i, j]] = "и",  
                Print[arr[[Mod[i + 1, 4], j]]]]]]]
```

р

24. Найти элемент массива, расположенный в строке справа от буквы, соответствующей номеру по списку в алфавите: $\Rightarrow \text{Array}[], \text{Position}[], \text{Part}[], \text{Mod}[]$.

```
In[825]:= For[i = 1, i ≤ 4, i++,  
            For[j = 1, j ≤ 8, j++,  
              If[arr[[i, j]] = "и",  
                Print[arr[[i, Mod[j + 1, 8]]]]]]]
```

й

НИУ МЭИ

Лабораторная работа №4

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

Преподаватель: Рытов А.А.

Москва 2016г.

1. Составить список, состоящий из первых четырех букв своей фамилии, программным путем найти порядковый номер каждой буквы в алфавите ("а"=1) и определить произведение этих порядковых номеров.

```

In[1]:= CharacterRange["А", "Я"]
Out[1]= {А, Б, В, Г, Д, Е, Ж, З, И, Й, К, Л, М, Н, О, П, Р, С, Т, У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ы, Ь, Э, Ю, Я}

In[2]:= Fam = Characters["якуш"]
Out[2]= {я, к, у, ш}

In[3]:= CodFam1 = ToCharacterCode[Fam]
Out[3]= {{1103}, {1082}, {1091}, {1096}}

In[4]:= CodFam = CodFam1 - 1072
Out[4]= {{31}, {10}, {19}, {24}}

In[5]:= Cd = Flatten[CodFam]
Out[5]= {31, 10, 19, 24}

In[6]:= pr = CodFam[[1]]
Out[6]= {31}

In[7]:= For[i = 2, i ≤ Length[CodFam], i++, pr = pr * CodFam[[i]]]

In[8]:= pr
Out[8]= {141360}

```

2. Сформировать простое число p1 с номером, соответствующим произведению четырех номеров из п.1

```

In[9]:= p1 = Prime[pr]
Out[9]= {1890277}

```

3. Ввести случайное целое число a, составляющее (90-N)% от величины простого числа, где N – номер по списку в группе.

```

In[10]:= a = IntegerPart[(90 - 13) * p1 / 100]
Out[10]= {1455513}

```

4. Найти число a^{-1} , обратное числу a по модулю p1, поочередно проверяя значения 1, 2, ... p1-1, пока не будет найдено $a^{-1} \pmod{p1}$, такое что $a * a^{-1} \equiv 1 \pmod{p1}$. Поиск выполнить используя операторы пакета «Математика»- Do, While, For (Built-in Functions \Programming \Flow Control). Вариант реализации определяется из следующего соотношения $nv = N \pmod{3} + 1$ и таблицы:

| | | | |
|----------------------|----|-------|-----|
| Номер варианта nv | 1 | 2 | 3 |
| Оператор | Do | While | For |

```

In[176]:= k = 1; While[k < p1 - 1, {If[Mod[a * k, p1] == 1, Print[k], None]}; k++]
943083

In[175]:= Do[{If[Mod[a * i, p1] == 1, Print[i], None]}, {i, 1, p1 - 1}]
943083

```

5. Провести проверку результата с помощью оператора Mod[].

```
In[178]:= Mod[a * 943 083, p1]
```

```
Out[178]= 1
```

6. Реализовать расширенный алгоритм Евклида, найти вектор “u” и число a^{-1} , обратное числу a по модулю p1.

```
In[51]:= u = {0, 1, p1}
```

```
v = {1, 0, a}
```

```
Out[51]= {0, 1, 1 890 277}
```

```
Out[52]= {1, 0, 1 455 513}
```

```
In[66]:= While[
```

```
u[[3]] ≠ 1,
```

```
q = IntegerPart[u[[3]] / v[[3]]];
```

```
t = u - q * v; u = v; v = t;
```

```
Print[u]
```

```
{-847 369, 652 474, 1}
```

```
In[68]:= nv2 = Mod[a * (-847 369), p1]
```

```
Out[68]= 1
```

7. Сравнить полученные результаты с результатом выполнения встроенной функции ExtendedGCD[p1,a].

```
In[57]:= ExtendedGCD[p1, a]
```

```
Out[57]= {1, {652 474, -847 369}}
```

8. Определить значение функции Эйлера для выбранного простого числа - EulerPhi[p1].

```
In[60]:= eu = EulerPhi[p1]
```

```
Out[60]= 1 890 276
```

9. Попробовать определить число a^{-1} , обратное числу a по модулю p1 путем прямого вычисления $a^{\phi(p1)-1} \bmod p1$.

```
In[61]:= Mod[Power[a, eu - 1], p1]
```

```
Out[61]= 1 042 908
```

10. Определить число a^{-1} , обратное числу a по модулю p1, используя функцию PowerMod[,,].

```
In[69]:= PowerMod[a, -1, p1]
```

```
Out[69]= 1 042 908
```


11. Используя функции `ClearSystemCache[]` (очистка системного кэша) и `Timing[]`, определить время поиска обратного элемента в п.4, п.6 и п.7.

```
In[78]:= ClearSystemCache[]
Timing[n = 1; While[Mod[a + n, p1] ≠ 1, n++]]

Out[79]:= {2.215214, Null}

ClearSystemCache[]
Timing[u = {0, 1, p1};
v = {1, 0, a};
While[u[[3]] ≠ 1, q = IntegerPart[u[[3]] / v[[3]]];
t = u - q * v;
u = v;
v = t];
t = u - v * q;
u = v;
v = t]

Out[81]:= {0., {-4 627 923, 3 563 500, 1}}
```

12. Определить списки делителей чисел $a-1$ и a , используя функцию `Divisors[]`. Найти одинаковые элементы в списках.

```
In[83]:= Divisors[a]
Out[83]:= {1, 3, 485171, 1455513}

In[88]:= a1 = 1042908
Out[88]:= 1042908

In[89]:= Divisors[a1]
Out[89]:= {1, 2, 3, 4, 6, 12, 233, 373, 466, 699, 746, 932, 1119, 1398, 1492, 2238, 2796, 4476, 86909, 173818, 260727, 347636, 521454, 1042908}
```

13. Разложить числа $a-1$ и a на простые множители, используя функцию `FactorInteger[]`. Проверить наличие одинаковых простых множителей.

```
In[90]:= FactorInteger[a]
FactorInteger[a1]

Out[90]:= {{3, 1}, {485171, 1}}

Out[91]:= {{2, 2}, {3, 1}, {233, 1}, {373, 1}}
```

14. Найти наибольший общий делитель чисел $a-1$ и a , для этого использовать функцию `GCD[]`.

```
In[92]:= GCD[a, a1]

Out[92]:= 3
```

15. Программируемым путем найти ближайшее, меньшее a , взаимно простое с a число и большее a взаимно простое с a число.


```
In[102]:= vsaim = a; While[GCD[a, vsaim] ≠ 1, vsaim++]; Print[vsaim]
          vsaim = a; While[GCD[a, vsaim] ≠ 1, vsaim--]; Print[vsaim]
          1455514
          1455512
```

НИУ МЭИ

Лабораторная работа №5

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

Преподаватель: Рытов А.А.

Москва 2016г.

1. Сформировать в «Блокноте» осмысленный текст на русском языке из N=30 букв, содержащий только строчные буквы и пробелы. Сохранить в виде текстового (*.txt) файла.
2. Запустить WinHex , открыть созданный файл, и с помощью программы PrintKey зафиксировать полученный результат (полный экран).
3. В меню Инструменты выбрать опцию Analise File и получить на экране распределение символов, содержащихся в выбранном файле. Зафиксировать (на произвольном носителе) те буквы и их количество , вероятность появления которых в выбранном тексте выше 0.05 (5%), например : E0h а 0.0795 96.
4. Запустить пакет "Математика" и создать список полученных в предыдущем пункте букв и их частот появления в виде: $ch0 = \{ \{ "a", 0.0795 \}, \{ "e", 0.0902 \}, \{ "и", 0.0637 \}, \{ "н", 0.0604 \}, \{ "о", 0.0762 \}, \{ "р", 0.0513 \}, \{ "т", 0.613 \}, \{ " ", 0.1358 \} \}$

```
ch0 = {{ " ", 0.1}, {"a", 0.667}, {"и", 0.1333}, {"к", 0.1333}, {"р", 0.1333}, {"т", 0.1}, {"y", 0.667}}
```

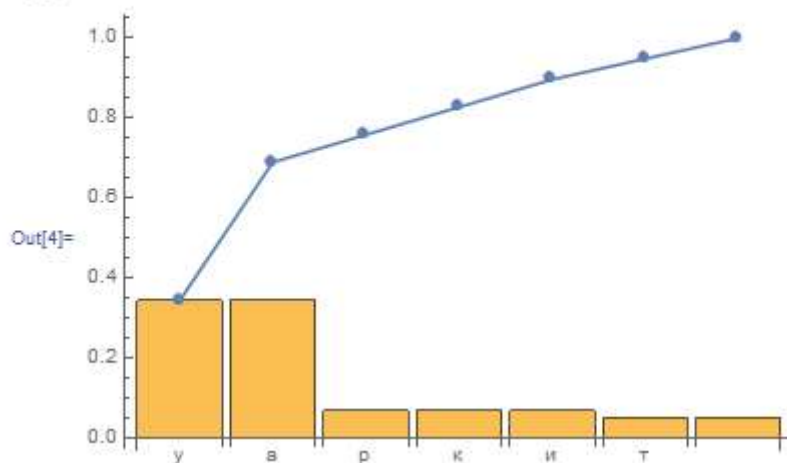
5. Нажав комбинацию клавиш Shift+Enter проверить правильность ввода (в дальнейшем любой запуск на выполнение операций сопровождается этой командой).

```
Out[117]= {{ , 0.1}, {a, 0.667}, {и, 0.1333}, {к, 0.1333}, {p, 0.1333}, {т, 0.1}, {y, 0.667}}
```

6. Подключить блок статистической обработки Needs["StatisticalPlots`"] и построить диаграмму Парето для введенного списка ParetoPlot[ch0].

```
In[3]:= Needs["StatisticalPlots`"]
```

```
In[4]:= ParetoPlot[ch0]
```



7. Вычислить длину списка (вектора) Length[ch0].

```
In[5]:= Length[ch0]
```

```
Out[5]= 7
```

8. Проверить сумму вероятностей элементов списка используя две операции: сначала создать список состоящий только из значений вероятности $p0=ch0[[All,2]]$, а затем подсчитать сумму вероятностей $summch0=Sum[p0[[i]],{i, Length[ch0]}]$.

```
In[6]:= p0 = ch0[[All, 2]]
```

```
Out[6]= {0.1, 0.667, 0.1333, 0.1333, 0.1333, 0.1, 0.667}
```

```
In[8]:= summch0 = Sum[p0[[i]], {i, Length[ch0]}]
```

```
Out[8]= 1.9339
```

9. Ввести список наиболее вероятных частот букв русского алфавита в виде $alfru=\{"a",0.062\},\dots\}$. Возможно использование файла alfru.doc.

Частоты букв p_i в русском языке

| | | | | | | | |
|--------|-------|---|-------|------|-------|---|-------|
| Пробел | 0,175 | р | 0,040 | я | 0,018 | х | 0,009 |
| о | 0,090 | в | 0,038 | ы | 0,016 | ж | 0,007 |
| е, ё | 0,072 | л | 0,035 | з | 0,016 | ю | 0,006 |
| а | 0,062 | к | 0,028 | ь, ъ | 0,014 | ш | 0,006 |
| и | 0,062 | м | 0,026 | б | 0,014 | ц | 0,003 |
| т | 0,053 | д | 0,025 | г | 0,013 | щ | 0,003 |
| н | 0,053 | п | 0,023 | ч | 0,012 | э | 0,003 |
| с | 0,045 | у | 0,021 | й | 0,010 | ф | 0,002 |

```
alfru = {{ " ", 0.175}, {"а", 0.062}, {"б", 0.014}, {"в", 0.038}, {"г", 0.013},
{"д", 0.025}, {"е", 0.072}, {"ж", 0.007}, {"з", 0.016}, {"и", 0.062},
{"й", 0.010}, {"к", 0.028}, {"л", 0.035}, {"м", 0.026}, {"н", 0.053},
{"о", 0.090}, {"п", 0.023}, {"р", 0.040}, {"с", 0.045}, {"т", 0.053},
{"у", 0.021}, {"ф", 0.002}, {"х", 0.009}, {"ц", 0.003}, {"ш", 0.006},
{"щ", 0.003}, {"ы", 0.016}, {"ь", 0.014}, {"э", 0.003}, {"ю", 0.006},
{"я", 0.018}}
```

10. Сформировать вектор частот $pa=alfru[[All,2]]$, определить длину списка $na=Length[pa]$ и проверить сумму вероятностей $summp=Sum[pa[[i]],{i,na}]$.

```

In[10]:= pa = alfru[[All, 2]]
na = Length[pa]
summp = Sum[pa[[i]], {i, na}]

Out[10]:= {0.175, 0.062, 0.014, 0.038, 0.013, 0.025, 0.072, 0.007, 0.016, 0.062,
0.01, 0.028, 0.035, 0.026, 0.053, 0.09, 0.023, 0.04, 0.045, 0.053, 0.021,
0.002, 0.009, 0.003, 0.006, 0.003, 0.016, 0.014, 0.003, 0.006, 0.018}

Out[11]:= 31

Out[12]:= 0.988

```

11. Рассчитать величину информационной энтропии H (энтропию языка):

$$H = -\sum_i P_i \cdot \log_2(P_i)$$

entropyalfru=N[-Sum[pa[[i]]*Log[2,pa[[i]]],{i,na}]].

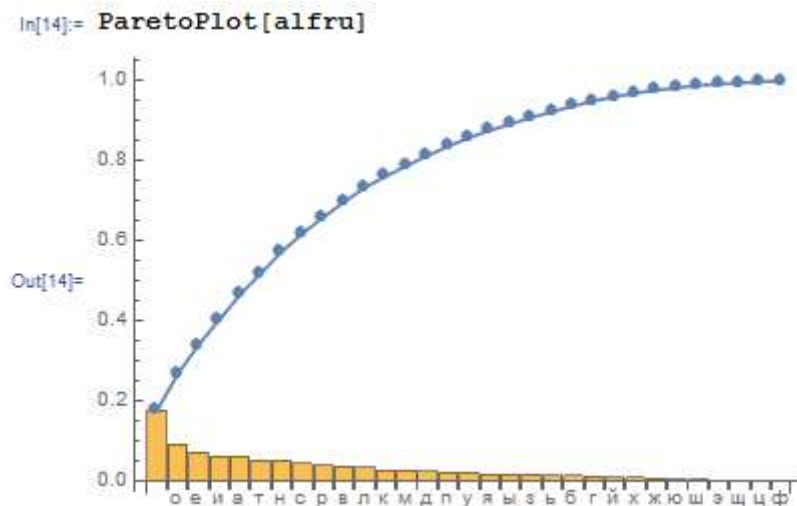
```

In[13]:= entropyalfru = N[-Sum[pa[[i]] * Log[2, pa[[i]]], {i, na}]]

Out[13]:= 4.27365

```

12. Построить диаграмму Парето для наиболее вероятного распределения букв русского языка в тексте.



13. Аналогично п.1 задания сформировать текстовый файл, содержащий 1500 строчных букв (и пробелов) русского алфавита.

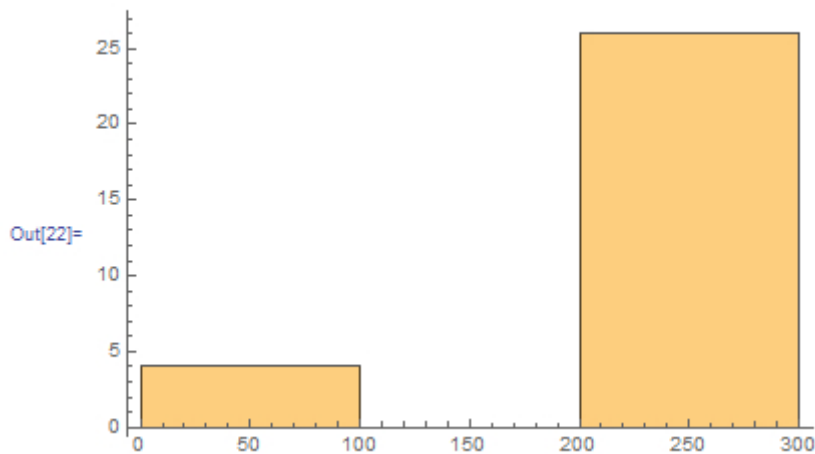
14. С помощью команды `v1 = ReadList["file", Byte, 30]` создать список данных `v1`, соответствующий сформированному текстовому файлу, где `file` - это полный путь к файлу, который можно ввести с помощью команд меню `Insert\ File Path`, 30 –число вводимых символов на первом этапе.

```
In[21]:= v1 = ReadList["H:\\МиСЭИ\\Lab 5\\text_1500.txt", Byte, 30]
```

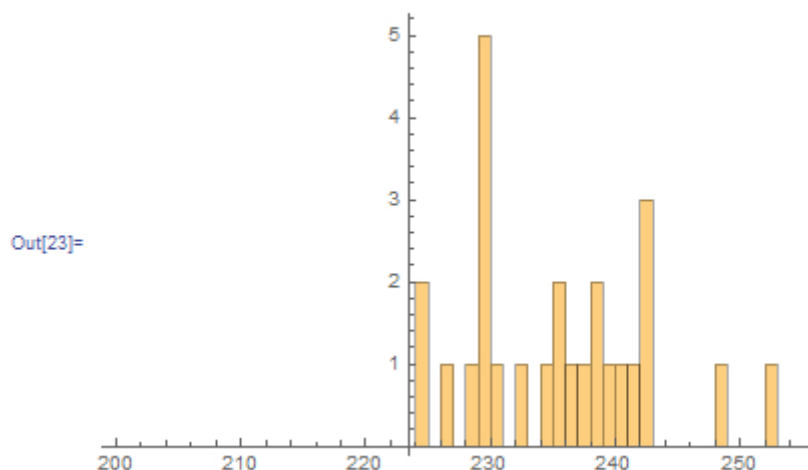
```
Out[21]:= {228, 224, 235, 229, 229, 32, 234, 235, 232, 229, 237, 242, 32, 236, 238,  
230, 229, 242, 32, 241, 238, 226, 229, 240, 248, 224, 242, 252, 32, 239}
```

15. Построить гистограммы распределения букв (символов), используя команды `Histogram[v1]`, `Histogram[v1, {200, 255, 1}]`.

```
In[22]:= Histogram[v1]
```



```
In[23]:= Histogram[v1, {200, 255, 1}]
```



16. Установить параметр `n2=60` и вести новый список данных `v2=ReadList["file1500",Byte,n2]`, где `file1500` –условное имя файла из п.13.

```
In[24]:= v2 = ReadList["H:\\МиСЭИ\\Lab 5\\text_1500.txt", Byte, 60]
```

17. Определить число символов, соответствующих буквам русского языка в векторе v2: freq2=Tally[v2].

```
In[25]:= freq2 = Tally[v2]
Out[25]:= {{228, 1}, {224, 5}, {235, 3}, {229, 9}, {32, 9}, {234, 1}, {232, 3},
           {237, 4}, {242, 4}, {236, 3}, {238, 3}, {230, 2}, {241, 2}, {226, 1},
           {240, 2}, {248, 1}, {252, 1}, {239, 2}, {251, 2}, {246, 1}, {243, 1}}
```

18. Создать список частот для введенных n2=60 символов текста:

p2=N[freq2[[All,2]/Length[v2]], где N[] – преобразование к действительной форме представления чисел.

```
In[26]:= p2 = N[freq2[[All, 2]] / Length[v2]]
Out[26]:= {0.0166667, 0.0833333, 0.05, 0.15, 0.15, 0.0166667, 0.05, 0.0666667,
           0.0666667, 0.05, 0.05, 0.0333333, 0.0333333, 0.0166667, 0.0333333,
           0.0166667, 0.0166667, 0.0333333, 0.0333333, 0.0166667, 0.0166667}
```

19. Определить длину списка p2, сумму вероятностей, и информационную энтропию.

```
In[29]:= lenp2 = Length[p2]
        sump2 = Sum[p2[[i]], {i, lenp2}]
        entropyalfrup2 = N[-Sum[p2[[i]] * Log[2, p2[[i]]], {i, lenp2}]]
Out[29]= 21
Out[30]= 1.
Out[31]= 4.01209
```

20. Подготовить список для записи энтропии 120 последовательно увеличивающихся сегментов подготовленного текста entropytextout=Range[120].

```
In[32]:= entropytextout = Range[120]
```

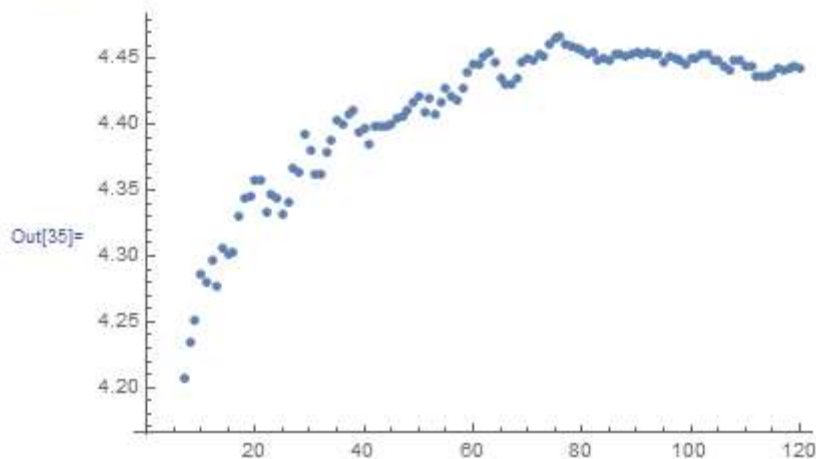
```
Out[32]:= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104,
105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120}
```

21. Используя оператор `Do[expr, {j, jmax}]`, построить вектор значений энтропии сегментов текста, увеличивающихся каждый раз на 10 символов:

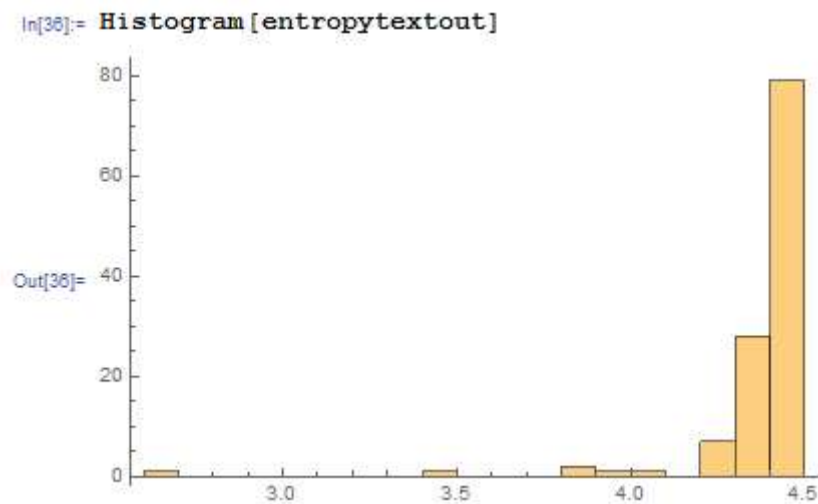
```
In[34]:= Do[{n2 = 10 * j;
v2 = ReadList["H:\\ММСЗИ\\Lab 5\\text_1500.txt", Byte, n2]; freq2 = Tally[v2];
p2 = N[freq2[[All, 2]] / Length[v2]]; np2 = Length[p2]];
entropytextout[[j]] = N[-Sum[p2[[i]] * Log[2, p2[[i]]], {i, np2}]], {j, 120}]
```

22. Построить точечный график зависимости энтропии сообщения от его длины: `ListPlot[entropytextout]`

```
In[35]:= ListPlot[entropytextout]
```



23. Построить гистограмму распределения рассчитанных значений энтропии.



24. Определить среднее значение $\text{Mean}[\text{entropytextout}]$ и дисперсию $\text{Variance}[\text{entropytextout}]$.

```
In[37]:= Mean[entropytextout]
          Variance[entropytextout]

Out[37]= 4.37255

Out[38]= 0.0424544
```

25. Используя команду $\text{Drop}[\text{list}, n]$ - возвращает список list, из которого удалены первые n элементов: - удалить из распределения явные выбросы (значения лежащие вне диапазона $\text{Mean}[\text{entropytextout}] \pm \text{Variance}[\text{entropytextout}]$) и вновь построить гистограмму распределения, рассчитать среднее значение и дисперсию энтропии.

```
In[69]:= max = Mean[entropytextout] + Variance[entropytextout]
          min = Mean[entropytextout] - Variance[entropytextout]

Out[69]= 4.415

Out[70]= 4.33009

In[80]:= len = Length[entropytextout]

Out[80]= 120

In[107]:= entropytextout = Sort[entropytextout]
```

```
Out[107]= {2.66644, 3.45418, 3.55378, 3.59474, 3.59108, 4.01209, 4.20725, 4.23488, 4.25054, 4.27673, 4.28062, 4.28657, 4.29657, 4.30169, 4.30574, 4.30881, 4.32987, 4.33229,
4.33346, 4.34099, 4.34469, 4.34891, 4.34505, 4.34672, 4.3579, 4.35829, 4.36262, 4.36904, 4.36444, 4.36668, 4.37966, 4.38041, 4.38525, 4.3879, 4.39249,
4.39489, 4.39494, 4.3982, 4.39859, 4.39902, 4.39974, 4.40055, 4.40345, 4.40438, 4.40631, 4.40765, 4.4079, 4.41003, 4.41055, 4.41149, 4.41657, 4.41782,
4.41859, 4.42059, 4.42099, 4.42229, 4.42836, 4.42836, 4.43053, 4.43154, 4.43559, 4.43581, 4.43712, 4.43723, 4.43767, 4.43814, 4.44031, 4.44132, 4.44192,
4.44251, 4.44291, 4.44304, 4.44403, 4.44431, 4.44471, 4.44495, 4.44607, 4.44647, 4.44672, 4.44715, 4.44804, 4.44822, 4.44848, 4.44869, 4.44931, 4.44903,
4.44929, 4.4494, 4.44938, 4.44984, 4.44996, 4.45025, 4.45038, 4.45072, 4.45087, 4.45219, 4.45246, 4.45249, 4.45276, 4.45302, 4.45307, 4.45309, 4.45318,
4.45326, 4.45351, 4.45405, 4.45421, 4.45428, 4.45442, 4.45452, 4.45456, 4.45473, 4.45552, 4.45698, 4.45607, 4.46043, 4.46161, 4.46183, 4.46384, 4.4674}
```

```
In[108]:= i = 1; While[entropytextout[[i]] < min, i++]; i
          j = 120; While[entropytextout[[j]] > max, j--]; j
```

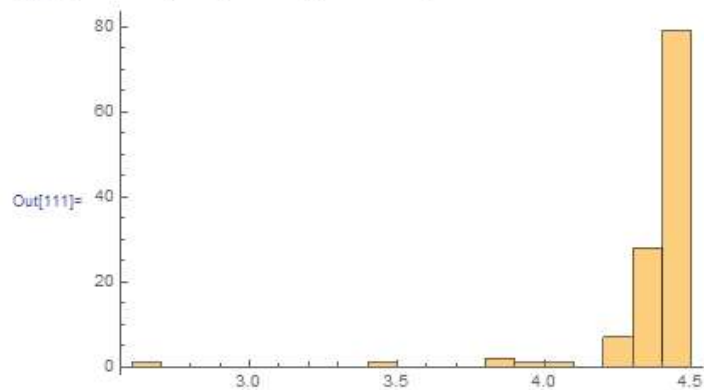
Out[108]= 18

Out[109]= 50

```
In[110]:= Drop[entropytextout, i]; Drop[entropytextout, -j]
```

```
Out[110]= {2.64644, 3.48418, 3.88378, 3.89474, 3.96108, 4.01209, 4.20725, 4.23488, 4.28054, 4.27673, 4.38062, 4.38607, 4.29607, 4.30189, 4.30374, 4.30681, 4.32987, 4.33229,
4.33949, 4.34090, 4.34363, 4.34891, 4.34805, 4.34672, 4.3579, 4.35829, 4.36262, 4.36304, 4.36444, 4.36468, 4.37966, 4.38041, 4.38526, 4.3879, 4.39249, 4.39469,
4.39694, 4.3982, 4.39959, 4.39902, 4.39974, 4.40055, 4.40345, 4.40435, 4.40631, 4.40765, 4.4079, 4.41003, 4.41055, 4.41149, 4.41657, 4.41752, 4.41853,
4.42059, 4.42099, 4.42229, 4.42836, 4.42835, 4.43053, 4.43154, 4.43559, 4.43551, 4.43712, 4.43723, 4.43767, 4.43814, 4.44031, 4.44152, 4.44182, 4.44281}
```

```
In[111]:= Histogram[entropytextout]
```



```
In[112]:= Mean[entropytextout]
          Variance[entropytextout]
```

Out[112]= 4.37255

Out[113]= 0.0424544

НИУ МЭИ

Лабораторная работа №6

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

Преподаватель: Рытов А.А.

Москва 2016г.

1. Для чистого Floppy – диска Для чистого Floppy – диска с помощью шестнадцатеричного редактора WinHex определить начало области форматирования по признаку (коду) F6F6 (WinHex => Инструменты (Tools) => Открыть диск (DiskEditor) => Physical Media(Physical Disk)=>Floppy disk 0(00h Floppy)=> OK).

| | | |
|----------|---|------------------|
| 00004200 | F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 | цццццццццццццццц |
| 00004210 | F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 | цццццццццццццццц |
| 00004220 | F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 | цццццццццццццццц |
| 00004230 | F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 | цццццццццццццццц |

2. С помощью стандартного текстового редактора «Блокнот», используя **латинский** алфавит, создать два тестовых файла с **разными названиями и разными текстами** (в пределах одного- двух предложений). Сохранить эти файлы в рабочей папке.
3. Запустить имеющийся файловый менеджер, скопировать файлы диск Н и с помощью редактора WinHex определить местонахождение (offset) имён файлов и местонахождение (offset) содержимого файлов.

| | | |
|----------|---|------------------|
| 00004200 | 43 68 69 6C 64 72 65 6E 20 6F 66 20 48 61 6E 6E | Children of Hann |
| 00004210 | 61 20 69 73 20 61 20 61 63 74 69 6F 6E 20 72 6F | a is a action ro |
| 00004220 | 6C 65 2D 70 6C 61 79 69 6E 67 20 67 61 6D 65 20 | le-playing game |
| 00004230 | 66 6F 72 20 74 68 65 20 6E 69 6E 74 65 6E 64 6F | for the nintendo |
| 00004240 | 20 44 53 20 68 61 6E 64 68 65 6C 64 2E 00 00 00 | DS handheld.... |
| 00004250 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00004260 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00004400 | 53 65 74 20 69 6E 20 68 69 67 68 20 66 61 6E 74 | Set in high fant |
| 00004410 | 61 73 79 2E 20 53 65 74 20 69 6E 20 68 69 67 68 | asy. Set in high |
| 00004420 | 20 66 61 6E 74 61 73 69 73 2E 00 00 00 00 00 00 | fantasis..... |
| 00004430 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00004440 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00004450 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00004460 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |

4. Стереть (Delete) файлы на дискете и закрыть файловый менеджер.
5. Запустить программу восстановления файлов PC Inspector File Recovery и убедиться в существовании файлов на дискете. Запомнить экран – PrintKey.



6. Запустить WinHex и через опции Search⇒ Find Text найти элементы имени файла и текста, записанного в «Блокноте». Зафиксировать изменения в структуре данных, происходящие при стирании информации.

| | | |
|----------|---|--------------------------|
| 00002600 | E5 41 42 20 20 20 20 20 54 58 54 20 18 12 5C 64 | eAB TXT ..\d |
| 00002610 | 85 49 85 49 00 00 3C 64 85 49 02 00 4D 00 00 00 | ...I...I...<d...I...M... |
| 00002620 | E5 41 42 42 20 20 20 20 54 58 54 20 18 6F B6 64 | eABB TXT .o\ld |
| 00002630 | 85 49 85 49 00 00 B4 64 85 49 03 00 2A 00 00 00 | ...I...I...rd...I...*... |
| 00002640 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00002650 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00002660 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00002670 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00002680 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00002690 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 000026A0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 000026B0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |

7. Повторить пункты 3 - 6 местонахождение (offset) рабочего задания, последовательно используя в качестве операции стирания информации с диска (зафиксировать область FAT) :

- удаление файла «перетаскиванием в корзину» - проверить наличие удаленного файла в «корзине»;

Нет удаленного файла в корзине

| | | |
|----------|---|--------------------|
| 00002600 | E5 52 49 4D 45 52 20 20 54 58 54 20 18 5A 62 63 | eRIMER TXT .Zbc |
| 00002610 | 7A 47 7A 47 00 00 39 63 7A 47 02 00 B4 00 00 00 | zGzG...9czG...r... |
| 00002620 | E5 45 58 54 20 20 20 20 54 58 54 20 18 5D 62 63 | eXT TXT .]bc |
| 00002630 | 7A 47 7A 47 00 00 42 63 7A 47 03 00 B8 00 00 00 | zGzG...BczG...ë... |

| | | |
|----------|---|------------------|
| 00004400 | 53 65 74 20 69 6E 20 61 20 68 69 67 68 20 66 61 | Set in a high fa |
| 00004410 | 6E 74 61 73 79 20 75 6E 69 76 65 72 73 65 20 43 | ntasy universe C |

| | | |
|----------|---|------------------|
| 00016896 | 43 68 69 6C 64 72 65 6E 20 6F 66 20 4D 61 6E 61 | Children of Mana |
| 00016912 | 20 69 73 20 61 20 20 61 63 74 69 6F 6E 20 72 6F | is a action ro |

➤ удаление файла Shift+Delete

| | | |
|----------|---|------------------|
| 00002600 | E5 52 49 4D 45 52 20 20 54 58 54 20 18 5A 2A 65 | eRIMER TXT .Z*e |
| 00002610 | 7A 47 7A 47 00 00 3A 63 7A 47 02 00 B4 00 00 00 | zGzG...czG..r... |
| 00002620 | E5 45 58 54 20 20 20 20 54 58 54 20 18 AB 2C 65 | eEXT TXT .«,e |
| 00002630 | 7A 47 7A 47 00 00 43 63 7A 47 03 00 B8 00 00 00 | zGzG..CczG..ë... |

| | | |
|----------|---|------------------|
| 00004400 | 53 65 74 20 69 6E 20 61 20 68 69 67 68 20 66 61 | Set in a high fa |
| 00004410 | 6E 74 61 73 79 20 75 6E 69 76 65 72 73 65 20 43 | ntasy universe C |

| | | |
|----------|---|------------------|
| 00016896 | 43 68 69 6C 64 72 65 6E 20 6F 66 20 4D 61 6E 61 | Children of Mana |
| 00016912 | 20 69 73 20 61 20 20 61 63 74 69 6F 6E 20 72 6F | is a action ro |

➤ быстрое форматирование диска H ;

| | | |
|----------|--|-------|
| 00002600 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |
| 00002610 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |

| | | |
|----------|---|------------------|
| 00004200 | 43 68 69 6C 64 72 65 6E 20 6F 66 20 4D 61 6E 61 | Children of Mana |
| 00004210 | 20 69 73 20 61 20 20 61 63 74 69 6F 6E 20 72 6F | is a action ro |

8. В блокноте набрать текстовый файл (размер – экран блокнота) из определенной последовательности символов, например – secret, и записать на чистый диск под именем abs. Определить число секторов, занимаемых файлом.

| | | |
|----------|---|-------------------|
| 00016896 | 73 65 63 72 65 74 73 65 63 72 65 74 73 65 63 72 | secretsecretse |
| 00016912 | 65 74 73 65 63 72 65 74 73 65 63 72 65 74 73 65 | etsecretsecretse |
| 00027840 | 73 65 63 72 65 74 73 65 63 72 65 74 73 65 63 72 | secretsecretse |
| 00027856 | 65 74 73 65 63 72 65 74 73 65 63 72 65 74 00 00 | etsecretsecret... |
| 00027872 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | |

22 сектора

9. Создать «короткий» текстовый файл (строка) и сохранить под тем же именем abs.
10. Используя WinHex исследовать размещение информации на диске – определить позицию размещения нового файла и секторы с содержимым предыдущего файла.

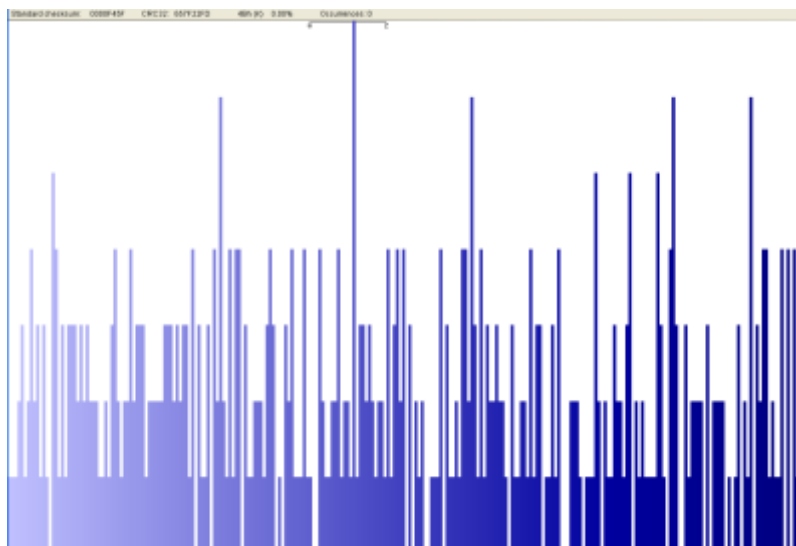
| | | | | | | | | | | | | | | | | | |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------------------|
| 00016896 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | secretsecretsecr |
| 00016912 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | etsecretsecretse |
| 00016928 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | crtsecretsecret |
| 00016944 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | secretsecretsecr |
| 00016960 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | etsecretsecretse |
| 00016976 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | crtsecretsecret |
| 00016992 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | secretsecretsecr |
| 00017008 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | etsecretsecretse |
| 00017024 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | crtsecretsecret |
| 00017040 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | secretsecretsecr |
| 00017056 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 73 | 65 | 63 | 72 | 65 | 74 | 00 | 00 | etsecretsecret.. |
| 00017072 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |

11.Отформатировать диск Н (см. п.1) и вновь записать на диск файлы, созданные в п.2

12.Провести безвозвратное стирание информации (первый файл п.2) с помощью утилиты eraser41(установить в случае необходимости): в режиме Pseudorandom Data при отключенных опциях меню Unused Disk Space– выбрать файл на диске, нажать правую кнопку мыши и выбрать опцию Erase.

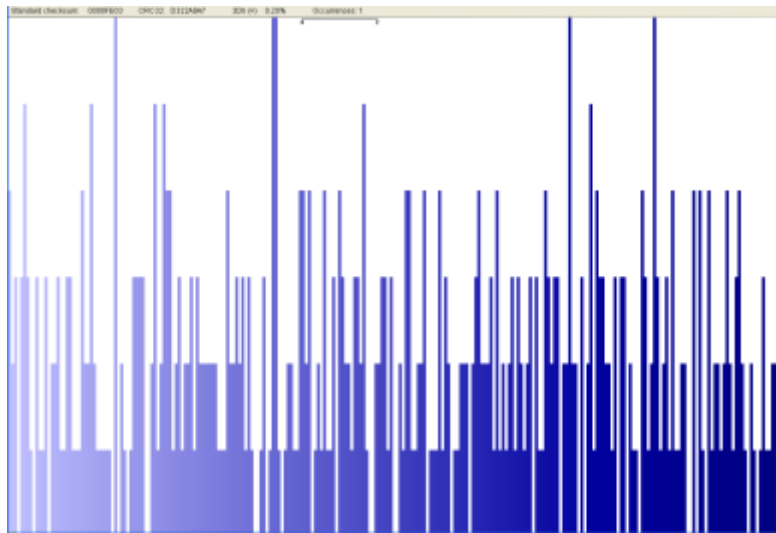
13. Используя WinHex исследовать размещение информации на диске:

выделив курсором стертый блок и, используя меню Правка\Сору Block, запомнить в отдельном файле содержимое этого блока; а также сохранить содержимое экрана с гистограммой распределения данных в блоке.

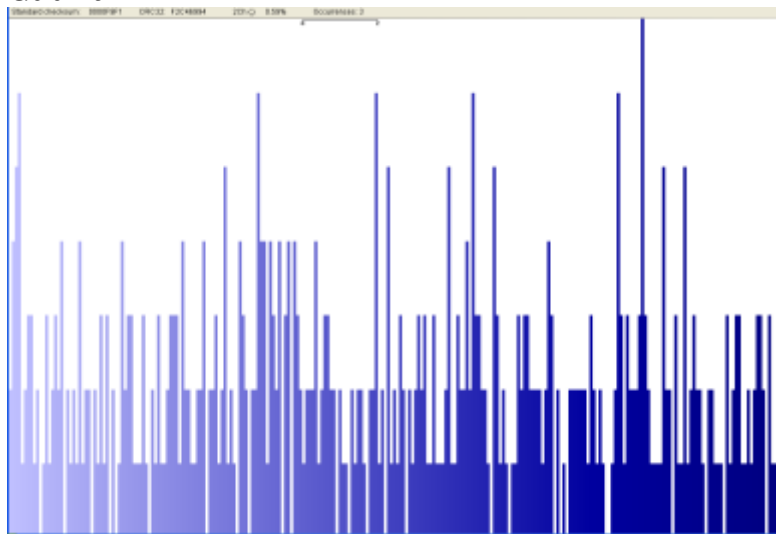


14.Отформатировать диск Н и вновь записать на диск файлы, созданные в п.2. Провести безвозвратное стирание информации (первый файл п.2) с помощью утилиты eraser41 в режимах Faster(Us DoD 5220.22-M) и Gutmann; для каждого режима повторить операции п.13.

Faster



Gutmann

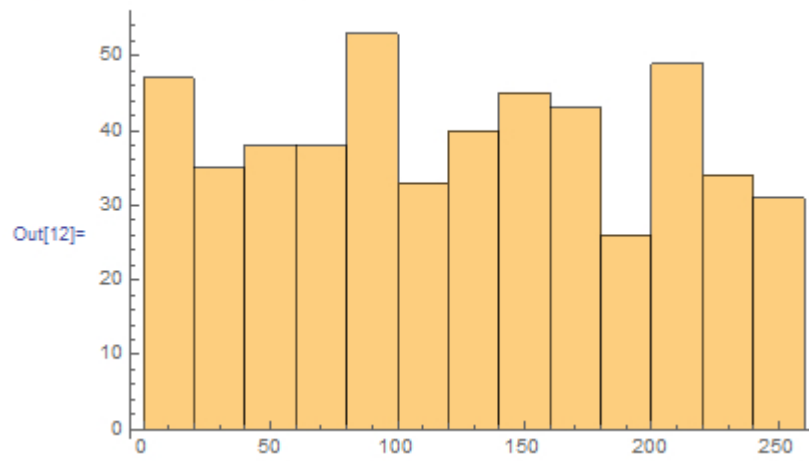


15. Определить энтропию стертого блока и построить гистограммы распределений псевдослучайных чисел для каждого из режимов. Для ввода данных использовать предварительное преобразование с помощью утилиты Converter.exe и последующей функции `ReadList["file.dat", Number]`.

```
In[15]:= v1 = ReadList["H:\1.dat", Number]
```



```
In[12]:= Histogram[v1]
```



НИУ МЭИ

Лабораторная работа №7

Выполнил: Якушенкова Ю. Е.

Группа: А-04-13

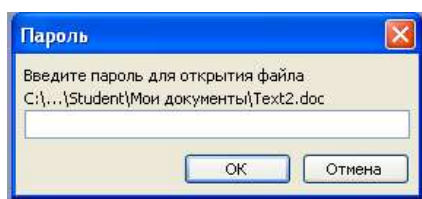
Преподаватель: Рытов А.А.

Москва 2016г.

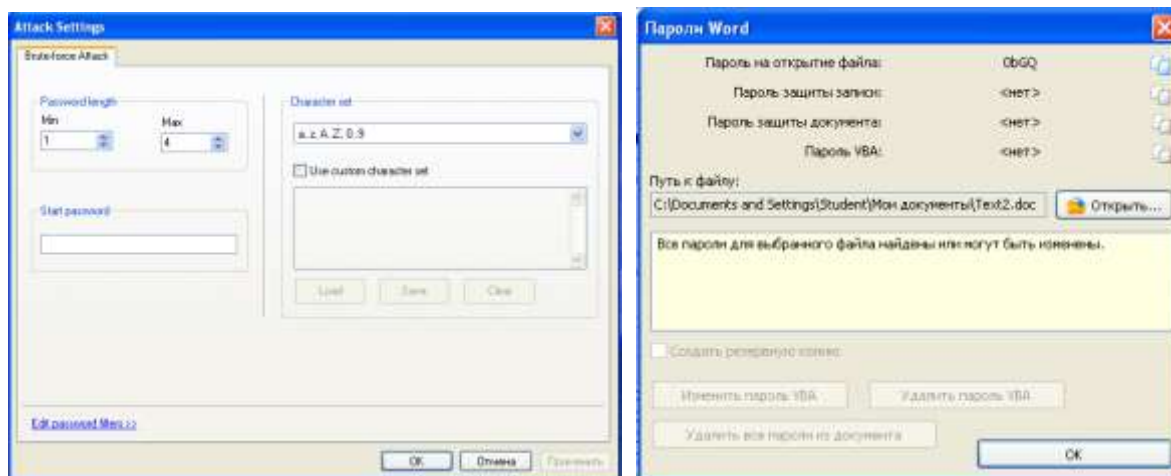
1. Разработать программную реализацию генератора паролей со следующими параметрами: длина пароля 4 символа, алфавит содержит латинские строчные буквы, латинские заглавные буквы, а также цифры. Сформировать 5 паролей для использования в лабораторной работе. Если по заданию длина пароля меньше 4-х символов, лишние символы отбрасываются.
2. В текстовом редакторе WinWord создать (открыть) 6 произвольных осмысленных текстов (300-400 символов) и сохранить их в 6-и разных файлах.
3. Для текста №1, в меню "Сервис\Защитить документ\Ограничения на редактирование" установить защиту – разрешить только чтение - по паролю: 4 символа пароля №1 из п.1.
4. Сохранить файл, закрыть и вновь его открыть.
5. Выделить весь текст или его часть и попробовать изменить текст, либо его удалить – результат зафиксировать.

Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1
Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1
Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1
Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1 Text1

6. Для текста №2. Установить пароль для открытия файла (**4 символа пароля №2 из п.1**) в файле WinWord (Файл\Сохранить как\Сервис\Параметры безопасности). После установки параметров безопасности файл сохранить и закрыть.
7. Попробовать открыть файл – результат зафиксировать.



8. Запустить Advanced Office Password Recovery (либо установить из Lab7\distributives\aoпр_setup_en.msi) и в меню "Опции" отключить все опции программы.
9. В меню "Атака" установить тип атаки "Атака перебором", в меню "Настройки Атак" установить максимальную длину пароля в 4 символа и набор символов для перебора (Character set) - "a..z,A..Z,0..9". Открыть защищенный файл с текстом №2. и запустить процесс восстановления паролей. Зафиксировать полученный результат: пароль и скорость перебора паролей, и открыть защищенный файл.



10. Вызвать архиватор 7-Zip File Manager и заархивировать 4 файла, используя следующие опции:

формат архива - **zip**;

метод сжатия - **LZMA**;

метод шифрования - **AES-256**;

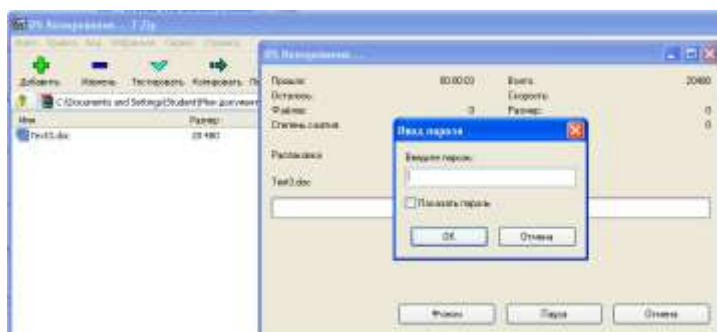
для текста №3 – пароль из 1-ой латинской строчной буквы;

для текста №4 – пароль из 2-х символов: 1 латинская строчная буква и 1 латинская заглавная ;

для текста №5 - пароль из 3-х символов: 1 латинская строчная буква, 1 латинская заглавная, 1 цифра ;

для текста №6 - пароль из 4-х символов: латинские строчные буквы, латинские заглавные буквы, все цифры ;

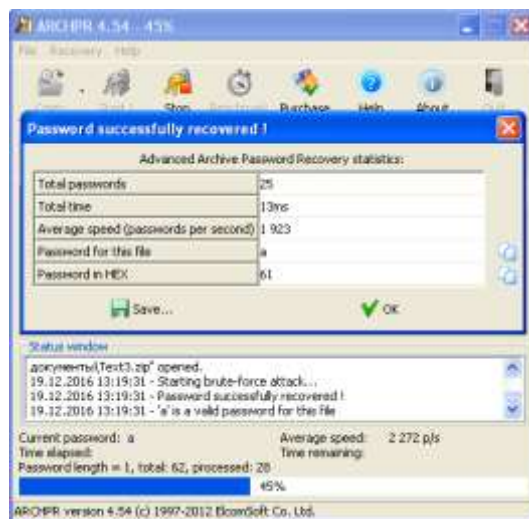
11. Закрывать 7-Zip File Manager и с помощью Total Commander (либо используя "Мой компьютер") попробовать просмотреть, а потом и открыть созданные архивы. Зафиксировать полученные результаты.



12. Запустить программу Advanced RAR Password Recovery (либо установить из Lab4\distributives\arpr.zip) и ввести следующие

параметры: набор - латинские строчные буквы, латинские заглавные буквы, все цифры ; тип атаки – перебор (Brute-force); длина пароля – минимальная 1,максимальная 4.

- 13.Определить пароль доступа к архивному файлу с текстом №3 и определить скорость перебора паролей (“Тест”).



14. Исследовать зависимость времени подбора пароля (минимальная длина=1, максимальная длина =4) в зависимости от его длины (1,2,3,4-символа) и при применении групп различных символов (меню «Набор» ARPR). При реализации атаки по маске полагаем,что один из 4-х символов пароля (например "S") и его позиция (3) известны , тогда маска имеет вид: ??S?.

| | Длина пароля | | | | |
|-------------------------|--------------------|--------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|
| | 1 | 2 | 3 | 4 | 4 с маской |
| Пароль | a | aB | aB1 | dA12 | dA12 |
| Набор символов | Строчные латинские | Строчные, заглавные латинские. | Строчные, заглавные латинские, цифры. | Строчные, заглавные латинские, цифры. | Строчные, заглавные латинские, цифры. |
| Скорость перебора, р/с | 1 923 | 5 953 | 6 081 | 4 288 | 4 972 |
| Время подбора расчетное | 13ms | 281ms | 17s 94ms | 26min 52s 363ms | 24s |

| | | | | | |
|---------------------------|----|-------|---------|------------|---------|
| Время подбора фактическое | 0 | 0 | 0 | 0 | 0 |
| Общее число паролей | 62 | 3 844 | 238 328 | 14 776 336 | 238 328 |
| Обработано паролей | 28 | 1 614 | 100 062 | 6 914 853 | 114 820 |

15. Определить ожидаемое время подбора пароля при силовой атаке, если пароль содержит символы фамилии и номера группы студента, а скорость подбора такая же, как в п.13.

s = ЯкушенковаЮлияА0413

$T_s = 0.5 \cdot A^s \cdot E \cdot \frac{1}{R} = 0.5 \cdot 76^{19} \cdot 1 \cdot \frac{1}{1923} = 1.4 \cdot 10^{32}$, где s – длина пароля; A – мощность алфавита; E – по сколько перебирает; R – скорость перебора