1. Последовательно определить значения вычетов по модулю 5 (Mod[***,5]) для чисел: 0,1,2,3,4,5,6,7,8,9,10.

```
Mod[{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, 5]
```

```
{0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0}
```

2. Провести операцию вычисления вычета по модулю 5 для списка list0 = Range[0,10].

```
list0 = Range[0, 10]
Mod[Range[0, 10], 5]
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
{0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0}
```

3. На множестве целых чисел создать четыре класса эквивалентности (списка) для операции Mod[***, pN],
где pN - простое число с номером Nсгр + 10, а Nсгр - номер по списку в группе:
list1 = {-2pN+1,....,- pN }; list2 = {-pN+1,...., 0 };
list3 = {0,...., pN-1 }; list4 = {pN,....,2pN-1 }.

```
pn = Prime[14]
list1 = Range[-2 * pn + 1, -pn]
list2 = Range[-pn + 1, 0]
list3 = Range[0, pn - 1]
list4 = Range[pn, 2 * pn - 1]
```

```
43
```

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

```
{-42, -41, -40, -39, -38, -37, -36, -35, -34, -33, -32, -31, -30,
 -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18, -17, -16,
 -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0}
```

```
{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, 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}
```

4. Привести по модулю pN элементы каждого из четырех списков, а затем отсортировать результаты (res1,res2,res3,res4) в порядке возрастания - Sort[***,Less].

```
res1 = Sort[Mod[list1, pn], Less]
res2 = Sort[Mod[list2, pn], Less]
res3 = Sort[Mod[list3, pn], Less]
res4 = Sort[Mod[list4, pn], Less]
```

```
{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, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42}

{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, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42}

{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, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42}

{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, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42}
```

5. Провести проверку эквивалентности списков res1,res2,res3,res4 двумя способами: а) при помощи оператора If[], логических функций сравнения == (Equal) и логического “и” && (And); Результат проверки вывести с помощью опертора Print[] в виде: “Списки совпадают” или “!!!Списки не совпадают”;
б) провести проверку с помощью функции Equivalent[].

```
If[res1 == res2 && res2 == res3 && res3 == res4 && res1 == res3 && res1 == res4 &&
  res2 == res4, Print["Списки совпадают"], Print["!!!Списки не совпадают"]]
Equivalent[res1, res2, res3, res4]
```

```
Списки совпадают
```

```
True
```

6. Для чисел a = 5+ Nгр , b=10+ Nгр , c=15+ Nгр и модуля pm = NextPrime[6+ Nгр] провести проверку эквивалентности следующих операций модулярной арифметики

In[21]:=
```
a = 9
b = 14
c = 19
pm = NextPrime[6 + 4]
Equivalent[Mod[a + b, pm], Mod[Mod[a, pm] + Mod[b, pm], pm]]
Equivalent[Mod[a - b, pm], Mod[Mod[a, pm] - Mod[b, pm], pm]]
Equivalent[Mod[a * b, pm], Mod[Mod[a, pm] * Mod[b, pm], pm]]
Equivalent[Mod[a * (b + c), pm], Mod[Mod[a * b, pm] + Mod[a * c, pm], pm]]
Equivalent[Mod[a^b, pm], PowerMod[a, b, pm]]
```

Out[21]= 9

Out[22]= 14

Out[23]= 19

Out[24]= 11

Out[25]= True

Out[26]= True

Out[27]= True

Out[28]= True

Out[29]= True

7. В конечном поле GF[pN] для числа s = Floor[pN /2] найти обратный элемент по сложению s', такой, что .

In[72]:=
```
pn = Prime[14]
s = Floor[pn / 2]
s2 = 0
For[i = 1, i ≤ pn, i++,
 If[Mod[s + i, pn] == 0, {s2 = i, Break[]}]
]
s2
```

Out[72]= 43

Out[73]= 21

Out[74]= 0

Out[76]= 22

Out[49]= 0

Out[51]= 0

8. В конечном поле GF[pN] для числа , найти число , обратное числу a по умножению

In[77]:=
```
pn = Prime[14]
a = Floor[pn / 3]
Rev[a0_, p0_] :=
 Module[{a = a0, p = p0},
  n = 1;
  a1 = -1;
  While[n < p - 1, If[Mod[a * n, p] == 1, a1 = n, a1 = a1]; n++];
  {a, a1, p}
 ]
Rev[a, pn]
```

Out[77]= 43

Out[78]= 14

Out[80]= {14, 40, 43}

9. Реализовать расширенный алгоритм Евклида, найти вектор “u” и число a-1, обратное по умножению к числу a по модулю pN. Для верификации алгоритма применить a = 5 и модуль p = 23. Программная реализация алгоритма должна быть выполнена в виде функции пользователя на основе оператора Module[] с двумя входными параметрами: a и p, и двумя выходными: a-1 и {u1,u2,u3}.

In[88]:=
```
Eucl[a0_, p0_] := Module[{a = a0, p = p0, u, v, q, t},
  u = {0, 1, p};
  v = {1, 0, a};
  While[u[[3]] ≠ 1,
   {
    q = IntegerPart[u[[3]] / v[[3]]],
    t = u - v * q,
    u = v,
    v = t
   }
  ];
      {u[[1]], u}
 ]
```

In[89]:=
```
Eucl[5, 23]
```

Out[89]= {-9, {-9, 2, 1}}

10. Сравнить полученные результаты с результатом выполнения встроенной функции ExtendedGCD[pN,a].

In[94]:=
```
ExtendedGCD[5, 23]
```

Out[94]= {1, {-9, 2}}

11. Определить значение функции Эйлера для модуля pN

In[95]:=
```
pn = Prime[14]
EulerPN = EulerPhi[pn]
```

Out[95]= 43

Out[96]= 42

12. Найти число , обратное числу a по модулю pN, путем прямого вычисления .

In[97]:=
```
pn = Prime[14]
a = Floor[pn / 3]
Mod[a^(EulerPN - 1), pn]
```

Out[97]= 43

Out[98]= 14

Out[99]= 40

13. Построить полную и приведенную систему вычетов по модулю , с использованием функции GCD[]. Определить число элементов полной и приведенной системы. Программная реализация алгоритма должна быть выполнена в виде функции пользователя на основе оператора Module[], входной параметр: r ,выходные параметры: список элементов полной системы, его длина, список приведенной системы вычетов, его длина. Сравнить полученные результаты с результатом вычисления функции EulerPhi[].

In[110]:=
```
pn = Prime[14]
r = Floor[3 * pn / 4]
SystemF[r_] :=
 Module[{r0 = r, full = {}, given = {}, i}, Print["Полная система вычетов"];
  full = Range[0, r0 - 1];
  Print[full];
  Print["Длина"];
  Print[Length[full]];
  given = {};
  For[i = 0, i ≤ Length[full], i++,
   If[GCD[full[[i]], r0] == 1, AppendTo[given, full[[i]]]]];
  Print["Приведенная система вычетов"] × Print[given];
  Print["Длина"];
  Print[Length[given]];
  Print["Длина EulerPhi "];
  Print[EulerPhi[r0]];]
SystemF[r]
```

Out[110]= 43

Out[111]= 32

Полная система вычетов

```
{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, 30, 31}
```

Длина

```
32
```

Приведенная система вычетов

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

Длина

```
16
```

Длина EulerPhi

```
16
```

14. Определить число , обратное числу a по модулю pN, используя функцию PowerMod [, ,].

In[114]:=
```
pn = Prime[14]
PowerMod[a, -1, pn]
```

Out[114]= 43

Out[115]= 40

15. Используя встроенную функцию Timing[], определить время поиска обратного элемента в п.8

Перед каждым измерением времени необходимо очистить системный кэш - ClearSystem-Cache[]. Сравнить время выполнения операции, выполненой по алгоритму п.8 со временем выполнения операции PowerMod[] для i=6.

In[123]:=
```
pn = Table[Prime[1 + 10^i], {i, 6}]
ClearSystemCache[]
Timing[PowerMod[a, -1, pn]]
```

Out[123]= {31, 547, 7927, 104 743, 1 299 721, 15 485 867}

Out[125]= {0.000012, {20, 508, 5096, 22 445, 835 535, 12 167 467}}

In[126]:=
```
ClearSystemCache[]
Timing[Rev[a, pn[[6]]]]
```

Out[127]= {15.2649, {14, 12 167 467, 15 485 867}}