In[19]:=
```
EllipticAdd[p_, a_, b_, c_, P_List, Q_List] ::
 Module[{lam, x3, y3, P3},
  Which[
   P == {0}, Q,
   Q == {0}, P,
   P[[1]] != Q[[1]],
        lam = Mod[(Q[[2]] - P[[2]]) PowerMod[Q[[1]] - P[[1]], p - 2, p], p];
        x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
        y3 = Mod[-(lam (x3 - P[[1]])) - P[[2]], p];
        {x3, y3},
   (P == Q) && (P[[2]] == 0), {0},
   (P == Q) && (P != {0}),
        lam = Mod[(3 P[[1]]^2 + 2 a P[[1]] + b) PowerMod[2 P[[2]], p - 2, p], p];
        x3 = Mod[lam^2 - a - P[[1]] - Q[[1]], p];
        y3 = Mod[-(lam (x3 - P[[1]])) - P[[2]], p];
        {x3, y3},
   P[[1]] == Q[[1]] && P[[2]] != Q[[2]], {0}]]
```

In[20]:=
```
MulByK[p1_, a1_, b1_, c1_, t_, n1_] ::
  Module[{p = p1, a = a1, b = b1, e = t, c = c1, n = n1, q1, q}, q = e;
   Do[{q1 = EllipticAdd[p, a, b, c, e, q], q = q1}, {i, 2, n}]; q];
```

In[21]:=
```
FindPoryadok[p1_, a1_, b1_, c1_, P1_] ::
 Module[{p = p1, a = a1, b = b1, c = c1, q1 = P1, q2 = P1, i}, i = 1;
  While[q2 != {0}, q2 = EllipticAdd[p, a, b, c, q1, q2]; i++];
  i]
```