NIS Lab-6

Q-Point generation on Elliptical curve

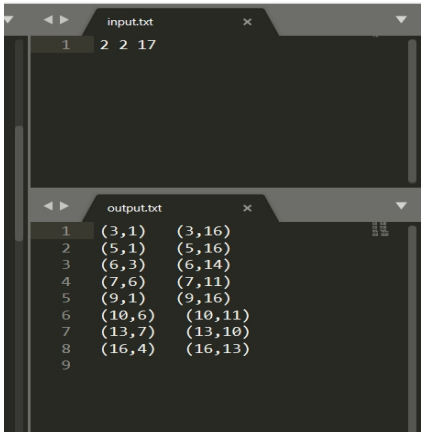
Code:

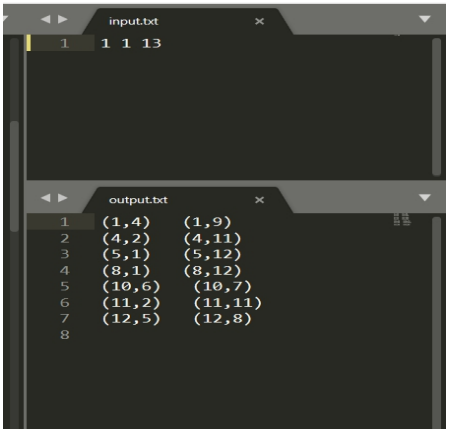
```
#include<bits/stdc++.h>
#include<iostream>
#include<cstring>
using namespace std;

#define int long long
#define pii pair<int,int>
#define vi    vector<int>

string toBinary(int n)
{
     string r;
     while (n != 0) {r = (n % 2 == 0 ? "0" : "1") + r;
n /= 2;}
     return r;
}

//calculate a^x mod n
int power(int a, int x, int n)
{
     int y = 1;
     string str = toBinary(x);
     reverse(str.begin(), str.end());
     for (int i = 0; i < str.length(); i++)
     {
          if (str[i] == '1')
          {
               y = (y * a) % n;
          }
          a = (a * a) % n;
     }
     return y;
}
```

```
int Evaluate_polynomaial(int x, int a, int b, int p)
{
    int temp1 = ((x * x) * x) + (a * x) + b;
    int ans = temp1 % p;
    return ans;
}

bool isPerfectSquare(int x)
{
    long double sr = sqrt(x);

    return ((sr - floor(sr)) == 0);
}

void Eliptic_curve_points(int a, int b, int p)
{
    int x = 1;
    int w;
    while (x < p)
    {
        w = Evaluate_polynomaial(x, a, b, p);
        //cout << w << endl;
        int pow = (p - 1) / 2;
        if (power(w, pow, p) == 1)
        {
            while (!isPerfectSquare(w))
            {
                w += p;
            }

            double a = sqrt(w);

            int a1 = (int)a;
            a1 = a1 % p;
            double b = (sqrt(w) * -1);
            int b1 = (int)b;
            b1 = b1 % p;

            while (b1 < 0)
            {
                b1 += p;
```

```
                }
                cout << "(" << x << "," << a1 << ")
(" << x << "," << b1 << ")" << endl;
                //return make_pair(a1, b1);
            }

            if (power(w, pow, p) == -1 or power(w, pow,
p) == (p - 1))
            {
                //cout << "No solution for x:" << x <<
" w:" << w << endl;
            }
            x++;
        }

}


int32_t main()
{
    #ifndef ONLINE_JUDGE
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
    #endif

    int a, b, p;
    cin >> a >> b >> p;


    Eliptic_curve_points(a, b, p);
    return 0;
}
```

Output:


input.txt
2 2 17
output.txt
(3,1) (3,16)
(5,1) (5,16)
(6,3) (6,14)
(7,6) (7,11)
(9,1) (9,16)
(10,6) (10,11)
(13,7) (13,10)
(16,4) (16,13)


input.txt
1 1 13
output.txt
(1,4) (1,9)
(4,2) (4,11)
(5,1) (5,12)
(8,1) (8,12)
(10,6) (10,7)
(11,2) (11,11)
(12,5) (12,8)