**Homework 6**

*Due Tuesday Nov 7th*

**1) (a)** Verify that the point P(0, 16) is on the elliptic curve y2 = x3 + 256, defined over the field of rational numbers Q.

**y2 = x3 + 256**

**P(0, 16)**

**x = 0**

**y = 16**

**162 = 03 + 256**

**256 = 256**

**P(0,16) is on the curve y2 = x3 + 256**

**(b)** Find 2P and 3P.

**1P = (0,16)**

**2P = (0,-16)**

**P = -2P**

**3P = O**

**(c)** Find the order of P (the smallest integer n such that nP = O).

**1P = (0,16)**

**2P = (0,-16)**

**P = -2P**

**3P = O**

**Order of P is 3**

**2)** Find the order of P(½, ½) over Q on the curve y2 = x3 + ¼x.

**1P = (0.5,0.5)**

**2P = (0,0)**

**4P = (0.5,-0.5)**

**1P = -4P**

**5P = O**

**Order of P is 5**

//  
// FILENAME: problem\_1a.cpp  
//  
  
#include <cstdlib>  
#include <iostream>  
  
// Verify that the point P(0, 16) is on the elliptic curve y^2 = x^3 + 256, defined over the field of rational numbers Q.  
  
int main()  
{  
 auto const x = 0;  
 auto const y = 16;  
  
 if (y\*y != x\*x\*x + 256) {  
 std::cout << "P(0,16) is NOT on the curve y^2 = x^3 + 256" << std::endl;  
 std::cout << y\*y << " != " << x\*x\*x << " + 256" << std::endl;  
 return EXIT\_FAILURE;  
 }  
  
 std::cout << "P(0,16) is on the curve y^2 = x^3 + 256" << std::endl;  
 std::cout << y\*y << " = " << x\*x\*x << " + 256" << std::endl;  
  
 return EXIT\_SUCCESS;  
}

//  
// FILENAME: problem\_1b.cpp  
//  
  
#include <cstdlib>  
#include <iostream>  
#include <utility>  
  
// Find the points 2P and 3P on the elliptic curve y^2 = x^3 + 256, defined over the field of rational numbers Q given P(0, 16).  
  
// alpha = ( 3 \* x[1] ^ 2 + a ) / ( 2 \* y[1] )  
// x[3] = alpha^2 − 2 \* x[1]  
// y[3] = -y[1] + alpha \* ( x[1] - x[3] )  
  
/\*\*  
 \* @brief Finds next point, 2P, on elliptic curve  
 \*  
 \* @param P given point, P  
 \* @param a slope of the line  
 \*  
 \* @return 2P  
 \*/  
std::pair<int, int> next\_point( std::pair<int, int> const& P, int a )  
{  
 int const x1 = P.first;  
 int const y1 = P.second;  
  
 int const alpha = ( 3 \* x1 \* x1 + a ) / ( 2 \* y1 );  
 int const x3 = alpha \* alpha - 2 \* x1;  
 int const y3 = -y1 + alpha \* ( x1 - x3 );  
  
 return std::make\_pair( x3, y3 );  
}  
  
/\*\*  
 \* @brief Given two points, P and Q, return the point, 2P, on an eliptic curve  
 \*  
 \* @param P the point P  
 \* @param Q the point Q  
 \*  
 \* @return 2P  
 \*/  
std::ostream& operator<<( std::ostream& os, std::pair<int, int> const& v )  
{  
 return os << "(" << v.first << "," << v.second << ")";  
}  
  
int main()  
{  
 int const x1 = 0;  
 int const y1 = 16;  
 int const a = 0;  
  
 auto const P1 = std::make\_pair( x1, y1 );  
  
 std::cout << "1P = " << P1 << std::endl;  
  
 auto P = P1;  
  
 for ( int i = 1 ; i < 2 ; ++i ) {  
 P = next\_point( P, a );  
 std::cout << i \* 2 << "P = " << P << std::endl;  
 }  
  
 return EXIT\_FAILURE;  
}

//  
// FILENAME: problem\_1c.cpp  
//  
  
#include <cstdlib>  
#include <iostream>  
#include <utility>  
  
// Find the order of P (the smallest integer n such that nP = O) on the elliptic curve y^2 = x^3 + 256, defined over the field of rational numbers Q.  
  
/\*\*  
 \* @brief Finds next point, 2P, on elliptic curve  
 \*  
 \* @param P given point, P  
 \* @param a slope of the line  
 \*  
 \* @return 2P  
 \*/  
std::pair<int, int> next\_point( std::pair<int, int> const& P, int a )  
{  
 int const x1 = P.first;  
 int const y1 = P.second;  
  
 int const alpha = ( 3 \* x1 \* x1 + a ) / ( 2 \* y1 );  
 int const x3 = alpha \* alpha - 2 \* x1;  
 int const y3 = -y1 + alpha \* ( x1 - x3 );  
  
 return std::make\_pair( x3, y3 );  
}  
  
/\*\*  
 \* @brief Print the given point  
 \*  
 \* @param os output stream object  
 \* @param v point object  
 \*  
 \* @return output stream  
 \*/  
std::ostream& operator<<( std::ostream& os, std::pair<int, int> const& v )  
{  
 return os << "(" << v.first << "," << v.second << ")";  
}  
  
int main()  
{  
 int const x1 = 0;  
 int const y1 = 16;  
 int const a = 0;  
  
 auto const P1 = std::make\_pair( x1, y1 );  
  
 std::cout << "1P = " << P1 << std::endl;  
  
 auto P = P1;  
  
 for ( int i = 1 ; i < 2 ; ++i ) {  
 P = next\_point( P, a );  
 std::cout << i \* 2 << "P = " << P << std::endl;  
 }  
  
 return EXIT\_FAILURE;  
}

//  
// FILENAME: problem\_2.cpp  
//  
  
#include <cstdlib>  
#include <iostream>  
#include <utility>  
#include <set>  
  
// Find the order of P(½, ½) over Q on the curve y^2 = x^3 + ¼x.  
  
/\*\*  
 \* @brief Finds next point, 2P, on elliptic curve  
 \*  
 \* @param P given point, P  
 \* @param a slope of the line  
 \*  
 \* @return 2P  
 \*/  
std::pair<float, float> next\_point( std::pair<float, float> const& P, float a )  
{  
 float const x1 = P.first;  
 float const y1 = P.second;  
  
 float const alpha = ( 3 \* x1 \* x1 + a ) / ( 2 \* y1 );  
 float const x3 = alpha \* alpha - 2 \* x1;  
 float const y3 = -y1 + alpha \* ( x1 - x3 );  
  
 return std::make\_pair( x3, y3 );  
}  
  
/\*\*  
 \* @brief Given two points, P and Q, return the point, 2P, on an eliptic curve  
 \*  
 \* @param P the point P  
 \* @param Q the point Q  
 \*  
 \* @return 2P  
 \*/  
std::pair<float, float> next\_point( std::pair<float, float> const& P, std::pair<float, float> const& Q )  
{  
 float const x1 = P.first;  
 float const y1 = P.second;  
  
 float const x2 = Q.first;  
 float const y2 = Q.second;  
  
 float const alpha = ( y2 - y1 ) / ( x2 - x1 );  
  
 float const x3 = alpha \* alpha - x1 - x2;  
 float const y3 = -y1 + alpha \* ( x1 - x3 );  
  
 return std::make\_pair( x3, y3 );  
}  
  
/\*\*  
 \* @brief Print the given point  
 \*  
 \* @param os output stream object  
 \* @param v point object  
 \*  
 \* @return output stream  
 \*/  
std::ostream& operator<<( std::ostream& os, std::pair<float, float> const& v )  
{  
 return os << "(" << v.first << "," << v.second << ")";  
}  
  
/\*\*  
 \* @brief Negate a point  
 \*  
 \* @param v input point  
 \*  
 \* @return the negated point (x, -y)  
 \*/  
std::pair<float, float> operator-( std::pair<float, float> const& v )  
{  
 return std::make\_pair( v.first, -v.second );  
}  
  
int main()  
{  
 float const x1 = 0.5;  
 float const y1 = 0.5;  
 float const a = 0.25;  
  
 auto const P1 = std::make\_pair( x1, y1 );  
  
 std::cout << "1P = " << P1 << std::endl;  
  
 auto const P2 = next\_point( P1, a );  
  
 std::cout << "2P = " << P2 << std::endl;  
  
 auto const P4 = next\_point( P1, P2 );  
  
 std::cout << "4P = " << P4 << std::endl;  
  
 auto P = P2;  
 auto Q = P4;  
  
 std::set<std::pair<float, float>> set{P1, P2, P4};  
  
 for ( int i = 3 ; true ; ++i ) {  
 auto const T = next\_point( P, Q );  
 P = Q;  
 Q = T;  
  
 std::cout << "2^" << i << "P = " << T << std::endl;  
  
 if ( set.find( -T ) != set.end() ) {  
 std::cout << "found additive inverse" << std::endl;  
 return EXIT\_SUCCESS;  
 }  
  
 set.insert( T );  
 }  
  
 return EXIT\_FAILURE;  
}