TEST 2

JASON MEDCOFF

1. Consider the curve $Y^2 = X^3 - 2X + 4$ and the points $P = (0, 2)$, $Q = (3, -5)$. Compute, by hand, $P \oplus Q$.

Solution. First, we obtain the equation of the line passing through P and Q as $y = -\frac{7}{3}x + 2$. Next, we want the intersections of this line with the curve, so set

$$\Big(-\frac{7}{3}x + 2\Big)^2 = x^3 - 2x + 4$$
$$0 = x^3 - \frac{49}{9}x^2 + \frac{22}{3}x$$

Clearly, $x = 0$ is a solution, so we divide by x.

$$0 = x^2 - \frac{49}{9}x + \frac{22}{3}$$

As we already know from point Q, $x = 3$ is also a solution so we use polynomial long division to obtain $x = \frac{22}{9}$. Then from the equation of the curve,

$$y^2 = (\frac{22}{9})^3 - 2(\frac{22}{9}) + 4$$

and we have that

$$y = \pm\sqrt{\frac{10000}{729}} = \pm\frac{100}{27}.$$

Plugging our $x = \frac{22}{9}$ into the line equation, we have the negative value, so the positive value gives the correct y for the new point. So the result of $P \oplus Q$ is $(\frac{22}{9}, \frac{100}{27})$.

2. Consider the cubic polynomial

$$x^3 + ax + b = (x - e_1)(x - e_2)(x - e_3)$$

and show that $4a^3 + 27b^2 = 0$ if and only if two or more of e_1, e_2, e_3 are the same.

Solution. Recall from elementary algebra that the discriminant is a function of a polynomial's coefficients, that gives interesting properties of the polynomial. Namely the discriminant equals zero if and only if the polynomial has a multiple root. So, we will begin by finding the discriminant of the given polynomial.

If we expand $(x - e_1)(x - e_2)(x - e_3)$, we have

$$x^3 - x^2(e_1 + e_2 + e_3) + x(e_1e_2 + e_1e_3 + e_2e_3) - (e_1e_2e_3)$$

but clearly it must be that $(e_1 + e_2 + e_3) = 0$. Then make the substitution $e_3 = -e_1 - e_2$. Then we obtain

$$x^3 + x(-e_1^2 - e_1e_2 - e_2^2) + (e_1^2e_2 + e_1e_2^2) \tag{1}$$

and it is plainly clear how a and b are expressed in terms of the roots.

The discriminant for a cubic with roots e_1, e_2, e_3 is given by

$$\Delta_f := \prod_{i \neq j}(e_i - e_j) = (e_1 - e_2)(e_2 - e_1)(e_3 - e_2)(e_2 - e_3)(e_1 - e_3)(e_3 - e_1)$$

and since $e_3 = -e_1 - e_2$, we can make the substitution

$$\Delta_f = (e_1 - e_2)(e_2 - e_1)((-e_1 - e_2) - e_2)(e_2 - (-e_1 - e_2))(e_1 - (-e_1 - e_2))((-e_1 - e_2) - e_1).$$

Expanding yields

$$\Delta_f = -4e_1^6 - 12e_1^5e_2 + 3e_1^4e_2^2 + 26e_1^3e_2^3 + 3e_1^2e_2^4 - 12e_1e_2^5 - 4e_2^6. \tag{2}$$

If we take $4a^3 + 27b^2$ and make the substitution for a and b shown in equation (1) in terms of the roots, we have

$$\begin{aligned} 4a^3 + 27b^2 &= 4(-e_1^2 - e_1e_2 - e_2^2)^3 + 27(e_1^2e_2 + e_1e_2^2)^2 \\ &= -4e_1^6 - 12e_1^5e_2 + 3e_1^4e_2^2 + 26e_1^3e_2^3 + 3e_1^2e_2^4 - 12e_1e_2^5 - 4e_2^6 \end{aligned}$$

which is precisely equation (2). Thus $4a^3+27b^2$ is the discriminant and it follows that $4a^3+27b^2 = 0$ if and only if two or more of e_1, e_2, e_3 are the same.

3. Consider

$$y^2 = x^3 + 2x + 3 \text{ over } \mathbb{F}_7$$

(a) List all the points on this curve.
Solution. The points on the curve are those that satisfy the equation in $\mathbb{F}_7$. That is, we can calculate $x^3 + 2x + 3$ for all x, then check if that number exists as a square in $\mathbb{F}_7$. Brute forcing, we have $y^2 \in \{3, 6, 1, 5, 0\}$ for $x \in \{0, 1, \ldots, 6\}$. In addition, we have $a^2 \in \{0, 1, 2, 4\}$ for $a \in \{0, 1, \ldots, 6\}$. Then the feasible y values are 0 and 1. Specifically, we have $(6, 0)$, $(2, \pm 1)$, and $(3, \pm 1)$.

(b) Make an addition table for these points.
Solution.

Table 1. Addition table

$\oplus$	$(6,0)$	$(2,1)$	$(2,-1)$	$(3,1)$	$(3,-1)$
$(6,0)$	$\mathcal{O}$	$(3,1)$	$(3,-1)$	$(2,1)$	$(2,-1)$
$(2,1)$	$(3,1)$	$(3,-1)$	$\mathcal{O}$	$(2,-1)$	$(6,0)$
$(2,-1)$	$(3,-1)$	$\mathcal{O}$	$(3,1)$	$(6,0)$	$(2,1)$
$(3,1)$	$(2,1)$	$(2,-1)$	$(6,0)$	$(3,-1)$	$\mathcal{O}$
$(3,-1)$	$(2,-1)$	$(6,0)$	$(2,1)$	$\mathcal{O}$	$(3,1)$

4. Consider the curve $y^2 = x^3 + x + 1$ over $\mathbb{F}_5$ and the points $P = (4, 2)$, $Q = (0, 1)$. Solve the discrete log problem, i.e. find n such that $Q = nP$.
Solution. Brute force all multiples of P using code. Clearly $n = 5$.

Table 2. Multiplication table

n	0	1	2	3	4	5	6	7	8
nP	$\mathcal{O}$	$(4,2)$	$(3,4)$	$(2,4)$	$(0,4)$	$(0,1)$	$(2,1)$	$(3,1)$	$(4,3)$

5. Consider the group of points on E. Since E is a finite group, every point in E has finite order. Then for some $P \in E$, if s is the smallest solution to $sP = \mathcal{O}$, s is the order of P. We know then that $(is)P = (i)(sP) = (i)\mathcal{O} = \mathcal{O}$ for all i. Then consider $nP = Q$. By the division algorithm and laws of exponents for groups,

$$\begin{aligned} nP &= Q \\ (is + r)P &= Q \\ (is)P \oplus (r)P &= Q \\ \mathcal{O} \oplus (r)P &= Q \\ (r)P &= Q \end{aligned}$$

but since r is less than s, it must be the smallest such solution to the equation; therefore $r = n_0$.

6. Use the double-and-add algorithm to compute nP in

$$y^2 = x^3 + 1828x + 1675 \text{ over } \mathbb{F}_1 999$$

for $n = 11$ and $P = (1756, 348)$. List the intermediate steps in a table.

Solution. We obtain the answer as $(1068, 1540)$. Table 3 displays the state of the intermediate point `r` in the code.

Table 3. Exponentiation of P

n	R
11	$(1756, 348)$
5	$(1756, 348)$
2	$(1362, 998)$
1	$(1362, 998)$
0	$(1068, 1540)$

7. Attach a listing of **your** code for

(a) Addition of points on an elliptic curve over a finite field.
(b) Exponentiation (nP).

Solution. Both pieces of code assume a point to be a 2-tuple. A curve is also a 2-tuple, given as (a, b) for $y^2 = x^3 + ax + b$.

```
def add(p1, p2, curve, p):
    if p1 is None:
        return p2
    if p2 is None:
        return p1
    if (p1[0] == p2[0]) and (p1[1] == (-p2[1]%p)):
        return None
    lam = 0
    if p1 == p2:
        lam = ((3*(p1[0]**2)+curve[0])*inverse_modp(2*p1[1], p)) % p
    else:
        lam = ((p2[1] - p1[1])*inverse_modp(p2[0] - p1[0], p)) % p
    x3 = (lam**2 - p1[0] - p2[0]) % p
    y3 = (lam*(p1[0] - x3) - p1[1]) % p
    return x3, y3
```

Note that the function `inverse_modp` finds the multiplicative inverse of its argument mod `p`; it is also homemade, from a past homework.

```python
def ec_exp(pt, n, curve, p):
    q = pt
    r = None
    while n>0:
        if (n % 2) == 1:
            r = add(r, q, curve, p)
        q = add(q, q, curve, p)
        n = n>>1
    return r
```