Directions: Solve the following problems. All written work must be your own. See the course syllabus for detailed rules.

1. Modular exponentiation in $\mathbb{F}_7$.

   (a) Fill in the table so that row a and column k contains a^k, where $a^k \in \mathbb{F}_7$.

a^k	0	1	2	3	4	5	6	7	$\cdots$
0	1	0	0	0	0	0	0	0	
1									
2									
3									
4									
5									
6									

   (b) For each non-zero element a, find the order of a in $\mathbb{F}_7^*$.

   (c) Use the table to find all primitive roots in $\mathbb{F}_7$. Verify that the number of primitive roots equals $\phi(6)$.

2. Use Fermat's Little Theorem and the fast power algorithm to compute the multiplicative inverse of 68 in $\mathbb{F}_{101}$.

3. Using as few modular exponentiation computations as possible, determine whether the following are primitive roots in $\mathbb{F}_{101}$.

   (a) 76

   (b) 87

   (c) 44

4. Primitive roots in $\mathbb{F}_{23}$.

   (a) Find a primitive root of $\mathbb{F}_{23}$.

   (b) Use part (a) to list all primitive roots of $\mathbb{F}_{23}$.

5. 🖳 Fast Power Algorithm.

   (a) In sage, reducing an integer a modulo m is achieved with the expression a % m. The following sage code implements the fast power algorithm, but a few blanks are missing. Fill in the blanks.

```
def fast_power (a, k, m):
    """for non-negative integers k, computes a^k modulo m"""
    if (k == 0):
        return ________

    if ((k % 2) == 1):
        # k is odd
        # recursively compute y = a^{k-1} (mod m)
        y = fast_power(a, k-1, m)

        return ________ % m
    else:
        # k is even
        # recursively compute y = a^{k/2} (mod m)
        y = fast_power(a, k/2, m)

        return ________ % m
```

   (b) Use the fast power algorithm to compute $(28324)^{14635515} \pmod{73254}$.

   (c) Use the fast power algorithm and Fermat's Little Theorem to make an educated guess about whether the following numbers are prime.

      i. 29399891
      ii. 34728343
      iii. 837249143
      iv. 296787375427