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

1. Modular Arithmetic Tables

   (a) Make addition and multiplication tables for $\mathbb{Z}_3$.

   (b) Make addition and multiplication tables for $\mathbb{Z}_6$.

2. Compute the following as efficiently as possible. Your answer should be an integer in the set $\{0, 1, \ldots, m-1\}$, where m is the modulus in the given problem.

   (a) $73 - 6173 \pmod{22}$

   (b) $342 \cdot 825 \pmod{17}$

   (c) $5 \cdot 6 \cdot 81 \pmod{83}$

   (d) $5^{20} \pmod{83}$

3. Use the extended Euclidean algorithm to compute the following inverses or show the inverse does not exist.

   (a) 3^{-1} in $\mathbb{Z}_{10}$.

   (b) 2^{-1} in $\mathbb{Z}_{10}$.

   (c) 38^{-1} in $\mathbb{Z}_{105}$.

4. Using a computer/calculator only for basic arithmetic operations (addition, subtraction, multiplication, and division), solve for x in $523x \equiv 211 \pmod{591}$. *Show your work.*

5. Let a, b, and m be integers.

   (a) Prove that if a and b both have inverses in $\mathbb{Z}_m$, then ab has an inverse in $\mathbb{Z}_m$.

   (b) Suppose that $pa + qm = 1$ and $rb + sm = 1$ for some $p, q, r, s \in \mathbb{Z}$. Find integers u and v such that $u(ab) + vm = 1$. [Hint: from $u(ab) - 1 = (-v)m$, implying $m \mid u(ab) - 1$ and hence $u(ab) \equiv 1 \pmod{m}$, what do you conclude about the relationship between u and ab in $\mathbb{Z}_m$?]

6. The recursive implementation of the extended Euclidean algorithm has some drawbacks. Usually, a programming language has a limited amount of space available to store the context of all recursive calls (this is called the *recursion stack*), and it is possible to run out stack space when a large input results in many nested recursive calls. If this happens in python, a `RecursionError` exception is raised. The function `recursiveEEA(a,b)` in the supplemental `inverse.py` file gives our familiar recursive implementation.

   There is a clever trick that exploits the associativity of matrix multiplication to give an iterative implementation of the extended Euclidean algorithm. In addition to avoiding overflow in the recursion stack, the iterative version has the advantage of using $O(\log a)$ space, whereas the recursive version may use up to $O(\log^2 a)$ space. The iterative implementation is not as easy to understand, however, and the code is less readable. An iterative implementation is provided in the function `EEA(a,b)` in the supplemental `inverse.py` file and follows:

```python
# returns (d,u,v) where d=gcd(a,b) and ua + vb = d
def EEA (a,b):
  ##
  ## Current status is the matrix [[A, B], [C, D]]
  ## matrix starts out as identity
  A = D = 1
  B = C = 0

  while (b > 0):
    q = a // b
    r = a - q*b

    ## update the matrix via [[A,B], [C,D]] = [[A,B],[C,D]]*[[0,1],[1,-q]]
    oldA = A
    oldC = C
    A = B
    C = D
    B = oldA - q*B
    D = oldC - q*D

    a = b
    b = r

  return (a, A, C)
```

(a) Implement the function `inverse(a,m)`, which returns the inverse of a in $\mathbb{Z}_m$ when the inverse exists and returns 0 if a has no inverse in $\mathbb{Z}_m$. If a has an inverse, the inverse returned should be in the set $\{1, \ldots, m-1\}$. For example, `inverse`$(8, 9)$ should return 8, not -1. Your implementation of `inverse(a,m)` should use an iterative implementation of the extended Euclidean algorithm, like `EEA(a,b)`. Give your code.

(b) Use your code to find the inverse of a in $\mathbb{Z}_m$, where a and m are the following integers.

$m =$ 8552688748587847369548628994659769019965171405583464208280713903011222
8526803677501505717544075197706813634147203744668225618518044671105397
6262134477710138712112945559140317733238199167152391972055479680971170
6842727571308047647532073780347697544430610959733844583335418192802795
3299286246608545269593078402940713478452246354730460617456070558236165
2006926458871481458182108312743460286036909677121395985848659773486557
0066962795744776600327038640336186795750843634237685340460839727144597
5322834936233612610653059714398623783493737255005796818491128677771151
532405933129565247405435807641391952414295952089262129 63

$a =$ 4674534506688880524568478689899693492312606098904224854024446247370881
1584717140255396165903745319923035052474049370974634324993822717332414
3022572629781420384204234509673008670701711946081387607701762025657370
8285899721977577700837025973234190575740127120965943098091458252134548
8043344222351602973116289543847819736285123505871596644829034586167933
7875723399602265065972276115707397884564055719642932359844934423134075
8770203510776856282054694562019935198367998034024549366811230471723716
5017455971699024156481267692801937401248384081000925252680858639979381
0972367692872988394644206860792040554682180609582166470 0

(c) What happens if you modify `inverse(a,m)` to use a recursive implementation of the extended Euclidean algorithm?