Challenge Problem #12: We've now learned two methods of computing modular inverses:

1. The extended Euclidean algorithm solves Bezout's identity to give the modular inverse.
2. Euler's thoerem provides an exponential form which gives the inverse, once the value of the totient function is known.

For this challenge problem, write a program that generates two random numbers a and b between 1 and 10,000. Your program should then (directly, not using libraries beyond basic mathematical operators (mod is allowed))

1. find the coefficient's in Bezout's identity.
2. find $\phi(a)$ and $\phi(b)$. (Note, your values are small enough to accomplish this by brute force easily) — If you want to be slick, first implement the Sieve of Eratosthenes to find all primes less than 10,000 and use them to quickly find the prime decompositions... (Many calculators and computer algebra systems contain these sorts of lookup tables).
3. if $\gcd(a, b) = 1$, find the modular inverses $a^{-1} \mod b$ and $b^{-1} \mod a$ using both the Bezout coefficients and by Euler's theorem.

Based on all your algorithm, which method is more efficient? Which was easier to implement? Include a short write-up of any interesting coding challenges you ran into and what you did to solve them.

Grading Criteria An 'M' will be based primarily on completion and delivering working code that accomplishes all of the above. An 'E' will be more thoughtful and polished in the final product. The author should determine reasonably efficient solutions to some of the steps, their code will be reasonably well documented with comments, and the final write-up will be up to professional technical writing standards.