Homework #2

1. Although ancient Chinese mathematicians did an excellent work inventing their remainder theorem, they sometimes made mistakes in their calculations. Since choosing prime numbers is an important part of the Chinese remainder theorem, they used a test for primality, which had flaws. The test outputs that $n$ is prime if and only if $n$ divides $(2^n-2)$.
   a. Give an example that satisfies the condition using an odd prime.
   b. Prove that the condition is true if $n$ is an odd prime.
   c. Give three examples of odd $n$ that is not a prime but divides $(2^n-2)$. This part demonstrates why ancient Chinese made occasional mistakes.

2. Alice designs a cryptosystems as follows (this system is due to Rabin). She chooses two distinct primes $p$ and $q$ (both $p$ and $q$ are congruent to 3 mod 4) and keeps them secret. She makes $n = pq$ public. When Bob wants to send Alice a message $m$, he computes $x \equiv m^2 \pmod{n}$ and sends $x$ to Alice. She have a decryption machine that does the following: When the machine is given a number $x$, it computes the square roots of $x$ mod $n$ since it knows $p$ and $q$. There is usually more than one square root. It chooses one at random, and gives it to Alice. When Alice receives $x$ from Bob, she puts it into her machine. If the output from the machine is a meaningful message, she assumes it is the correct message. If it is not meaningful, she puts $x$ into the machine again. She continues until she gets a meaningful message. Assume Bob sends a meaningful message.
   a. Why should Alice expect to get a meaningful message fairly soon? Give a probabilistic analysis of why the chance of getting the correct meaningful message increases with each trial of Alice.
   b. If Oscar intercepts $x$ (he already knows $n$), why should it be hard for him to determine the message $m$?
   c. If Eve breaks into the Alice’s office and thereby is able to try a few chosen ciphertext attacks on Alice’s decryption engine, how can she determine the factorization of $n$?

3. The extended Euclidean algorithm can be adapted to find the multiplicative inverse of a polynomial. Specifically, the algorithm will find the multiplicative inverse of $a(x) \mod f(x)$ if the degree of $a(x)$ is less than the degree of $f(x)$ and $\gcd(a(x), f(x)) = 1$. If $f(x)$ is an irreducible polynomial, then it has no factor other than itself and 1, so that $\gcd(a(x), f(x)) = 1$. The algorithm, which can be used to calculate inverses in binary extension fields $GF(2^n)$, is given in the Appendix. Use this algorithm to compute the multiplicative inverse of $(x^7 + x^6 + 1) \mod (x^8 + x^4 + x^3 + x + 1)$. (This irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ is used in AES).
   **Caveat:** Show the steps in your calculations. As an alternative, you can code the algorithm; however, it is not allowed to use direct computation techniques provided by Matlab, Maple, Mathematica, etc. You should implement the polynomial addition, multiplication and division operations in order to implement the inversion.

4. What other algorithm can you think of for computing inverse in $GF(2^n)$?
APPENDIX

The following algorithm is given for general case. But we are working with binary polynomials. Therefore, keep in mind subtraction is the same as the addition.

2.221 Algorithm Extended Euclidean algorithm for $\mathbb{Z}_p[x]$

INPUT: two polynomials $g(x), h(x) \in \mathbb{Z}_p[x]$.
OUTPUT: $d(x) = \gcd(g(x), h(x))$ and polynomials $s(x), t(x) \in \mathbb{Z}_p[x]$ which satisfy $s(x)g(x) + t(x)h(x) = d(x)$.

1. If $h(x) = 0$ then set $d(x) \leftarrow g(x)$, $s(x) \leftarrow 1$, $t(x) \leftarrow 0$, and return $(d(x), s(x), t(x))$.
2. Set $s_2(x) \leftarrow 1$, $s_1(x) \leftarrow 0$, $t_2(x) \leftarrow 0$, $t_1(x) \leftarrow 1$.
3. While $h(x) \neq 0$ do the following:
   3.1 $q(x) \leftarrow g(x) \div h(x)$, $r(x) \leftarrow g(x) - h(x)q(x)$.
   3.2 $s(x) \leftarrow s_2(x) - q(x)s_1(x)$, $t(x) \leftarrow t_2(x) - q(x)t_1(x)$.
   3.3 $g(x) \leftarrow h(x)$, $h(x) \leftarrow r(x)$.
   3.4 $s_2(x) \leftarrow s_1(x)$, $s_1(x) \leftarrow s(x)$, $t_2(x) \leftarrow t_1(x)$, and $t_1(x) \leftarrow t(x)$.
4. Set $d(x) \leftarrow g(x)$, $s(x) \leftarrow s_2(x)$, $t(x) \leftarrow t_2(x)$.
5. Return $(d(x), s(x), t(x))$.

Or see the Handbook of applied cryptography book, Page 82. Follow the URL http://www.cacr.math.uwaterloo.ca/hac/about/chap2.pdf