CS547A 2003 Homework set #1

Due Friday September, 26 2003 in class at 13:30 SHARP

Solve (by hand calculations) the following system of congruences:

$$x \equiv 13 \pmod{35}$$
$$x \equiv 11 \pmod{36}$$
$$x \equiv 23 \pmod{37}.$$

(from Brassard-Bratley's book)

8.5.13 Let $p \equiv 1 \pmod 4$ be a prime, and let **x** be in QR_p.
An integer **a**, $0<a<p$, *gives the key* to $\sqrt{x}$ if $(a^2-x) \bmod p$ is in QNR_p.
Prove that

a) Algorithm **rootLV** finds a square root of **x** if and only if it randomly chooses an integer **a** that gives the key to $\sqrt{x}$.

b) Exactly $(p+3)/2$ of the $p-1$ possible random choices for **a** give the key to $\sqrt{x}$.

Consult handout for appropriate HINT.

More Exercises

1. Assume that the prime number theorem is exactly correct for all powers of two. Calculate a good upper bound on the probability that we mistakenly output a composite number instead of a prime after the following events have occurred:

- pick a random **m**-bit integer **n** such that $\gcd(n,6)=1$
- *(also explain an easy way to do the above efficiently)*
- the procedure **Rabin-Miller prime(n,k)** returns 'prime'

a) Express your bound as a function of **m** and **k**.

b) If I want a random **512** bits prime **p**, what value of **k** should be used in **Rabin-Miller prime(p,k)** to guarantee probability at most $1/2^{20}$ of outputting a composite number?

... more on back

2. Jacobi Symbol Algorithm. Let's use the following names for the six properties of the Jacobi Symbols as given in Sec 1.5 of the class notes:

1-prop, mul-prop, mod-prop, -1-prop, 2-prop, inv-prop

a) Justify each part of Algorithm 1.2 using these properties.

b) Show that when computing **(a/n)**, for odd n, after any two recursions of the algorithm, the total size in bits of the numbers **(|a|+|n|)** involved has decreased by at least one.

3. MAPLE™.

a) Write a MAPLE function **pqsqrt(x,p,q)** similar to **msqrt** that receives three integers **x,p,q** such that the latter two are primes. Your function should return a square root of **x** modulo **n=p*q** or **FAIL** if no such square root exists. Don't forget the case **p=q**. (also test that **p** and **q** are indeed primes, else return **FAIL**) Why do we need this new function instead of **msqrt** ?

b) Pick at random two primes **p,q** of 100 digits each, both of which start with a one followed by your **student ID number** with **p** ending in **01** and **q** ending in **03**. Compute **n=p*q**. Obvisouly, this method promises that you each have a distinct **n**.

c) Pick at random two quadratic non-residues **y** and **z** such that **(y/n)=+1** and **(z/n)=-1**.

d) For each number **x** in the range **[1234567890,…,1234567989]** decide whether it is a quadratic residue **mod n** or not and justify your decision by exhibiting a square root **mod n** of one of the following possibilities **x, yx, zx, zyx**. Summarize your results in a table of the number of elements of each type you found. Example:

Type	x	yx	zx	zyx
Amount	21	27	24	28

e) Explain how we could check that you accomplished parts **b,c)** correctly without asking you **p** and **q** (given only **(n, y, z)** and the anwers of part **d)**).

… more on back