CS 667 : Homework 2(Due: Oct 7, 2009)

Problem 1. (50 POINTS)
Evaluating a polynomial A(x) of degree-bound n at a given point x_0 can also be done by dividing $A(x)$ by the polynomial $(x - x_0)$ to obtain the quotient polynomial $q(x)$ of degree-bound $n - 1$ and a remainder r, such that

$$A(x) = q(x)(x - x_0) + r$$

Clearly $A(x_0) = q(x_0)(x_0 - x_0) + r = r$. Show how to compute the remainder r and the coefficients of $q(x)$ in time $\Theta(n)$ from x_0 and the coefficients $a_0, a_1, \ldots,$ of A.

Problem 2. (50 POINTS)
Show that interpolation can be done in $O(n^2)$ time.
Hint: Read Exercise 30.1-5 of CLRS on page 830, and think of the implications of Problem 1 of this homework and Problem 4 of HW1.

Problem 3. (50 POINTS)
(a) We want to compute the product $m = m_1 m_2 \ldots m_k$. Show that m can be computed in $O((\lg m)^2)$ bit operations.
(b) Use part (a) to show how to compute $n!$ (n factorial) in $O((n \lg n)^2)$ bit operations.
(c) Given integers a and b how long (in bit, NOT word operations) does it take to efficiently compute integer $n = a^b$, i.e. raise a to the power of b? Express the number of bit operations required in asymptotic notation as a function of n in an efficient solution to this problem. **Note**. An answer of the form: "*At most* $2 \lg n - 1$ *multiplications*" is not suitable as an answer to this problem.

Problem 4. (50 POINTS)
Show that the GCD computation requires $O(\lg a \lg b)$ bit operations thus improving the rough $O(\lg^3 a)$ bound given in class, i.e. complete the proof of Corollary 5 of the notes (Subject 2, page 32.

Problem 5. (50 POINTS)
(a) Derive a point value representation for $A^{rev}(x) = \sum_{j=0}^{n-1} a_{n-1-j} x^j$ from a point-value representation for $A(x) = \sum_{j=0}^{n-1} a_j x^j$, assuming that none of the points in the point-value representation of $A(x)$ is 0.
(b) What is the largest k such that if you can multiply 3×3 matrices using k multiplications (not assuming commutativity of multiplication), then you can multiply $n \times n$ matrices in time $o(n^{\lg 7})$? What would the running time of this algorithm be?

Problem 6. (50 POINTS)

Implement Shamir's secrete sharing scheme.

```
ShamirCreate(secret k, parties p, reconstruct  r, file-out file-name)
  // Returns  a file-name that contains one per-line the individual secrets
  // assigned to each of the p parties. file-name thus has p lines one for each party

secret ShamirReconstruct(parties p, reconstruct  r, file-in  file-name)
  // Uses file-name with at least r lines but no more than p to reconstruct
  // the secret s that is returned.

  // Details are left to you for implementation.
```

The interface will be through the command-line. A

```
% ./ShamirCreate k p r out-file
```

will call the corresponding function and generate some output in file out-file. (Note that `ShamirCreate` is not only a function name but also a program name.)

A `ShamirReconstruct p r my-file` will use my-file (containing lines of out-file) to return in the standard output the secret.

The catch of this Problem: Numbers can grow very big! The secret is a positive 32-bit integer `int` or `unsigned int`.

Problem 7. (100 POINTS)

Implement the Karatshuba-Ofman algorithm. The input/output will be decimal (base-10) integers. Whether you plan to maintain the long integers in binary or not it's up to you.

```
// The following is pseudocode
// Whether you provide an interface in which n3 is an integer, a pointer to an
// integer or a reference it's up to you.
ReadKaratsuba(file digit1, longint d1, int n1);
MultiplyKaratuba(longint d1, int n1,longint d2, int n2, longint d3, int n3);
WriteKaratsuba(longint d1, int n1, file digit1);
```

A file contains on its first line an (in fact two) integers indicating the number of (decimal) digits that will follow. The remaining lines contain the digits. Line breaks, spaces, tab might exist but are ignored. For example

```
10 10
12345 56
789
```

stores integer the 10-digit integer `1234556789`. Note that the length in digits is given twice in the first line. The first integer is indeed the length; the second is to note the decimal-point in the case that the number is not an integer. Thus viewing the input as real number we get `1234556789.` by including a decimal point to the right of the 10-th digits (the second 10 of the first line).

I should be able to test your code from the command line with an operation of the form

```
% ./karatshuba  file1 file2 file3
```

where `file1, file2` contain the two integers that will be multiplied and `file3` will store the product in the same format as that described earlier.

The first function reads a long integer from a file. The second function uses the internal representation and the algorithm presented in class for binary numbers (adapt it as fit) to speed up multiplication faster than $\Theta(n^2)$. The last function prints an internally-stored integer into a file in the standard format described in this assignment. Make sure that in the latter case you never print more that 50 decimal digits per line.

Note. Java has extensive `BigInteger` capabilities that might include a Karatsuba implementation. Those of you who plan to use Java, you are not allowed to use this functionality, i.e. you need to implement functions from scratch, i.e. you can not even use the `BigInteger` class. Similarly for C/C++ implementors, additional libraries (eg gmp or the like) are not allowed.