Design and Analysis of Algorithms 2011
(Home work 2)

September 20, 2011

- All problems except problem 15 is due on Monday, October 3, before 10 a.m. See problem 15 for submission instructions of that problem.
- Late home works will not be accepted.
- Please give precise arguments for all statements that you write.
- Please do not hesitate to contact me if you do not understand the problems.
- Collaboration is discouraged, but not prohibited. It is recommended that you try to solve the problems on your own. You can discuss the questions with your colleagues but you should not copy solutions. Always write down your own answers. If copying is detected that may immediately lead to a grade less than 7. (**This would be followed strictly**)
- Credits would be given to partial solutions also.
- When you write an algorithm, you should briefly discuss state and prove the running tthe main idea of your algorithm, then write a pseudo code, argue about its correctness and running time of your algorithm.
- The answers (except for problem 15) should be typed or written clearly and a hard copy is to be submitted.

Playing With Numbers: [$5 \times 10 = 50$ **points**]

1. If $\gcd(m, n) = 1$, given a and b, prove that there exists an x such that $x \equiv a \bmod m$ and $x \equiv b \bmod n$.
2. Let n be a positive composite integer. Show that there exist a prime p dividing n such that $p \leq \sqrt{n}$.
3. Show that for any integers a and b with $d = \gcd(a, b) \neq 0$, we have $\gcd(\frac{a}{d}, \frac{b}{d}) = 1$.

4. Let $a, b, n_1, n_2 \in \mathbb{Z}$ with $n_1 > 0$ and $n_2 | n_1$. Show that if $a \equiv b \bmod n_1$ then $a \equiv b \bmod n_2$.

5. Let $a, b, n \in \mathbb{Z}$, such that $n > 0$ and $a \equiv b \bmod n$. Show that $\gcd(a, n) = \gcd(b, n)$.

6. Find $d = \gcd(1024, 888)$ and find $x, y \in \mathbb{Z}$ such that $d = 1024x + 888y$.

7. If F_n be the n-th Fibonacci number, show that for any $n \geq 1$, $\gcd(F_{n+1}, F_n) = 1$.

8. What is $2^{2^{2009}} (\text{mod} 3)$?

9. Is the difference of $5^{30,000}$ and $6^{123,456}$ a multiple of 31?

10. Calculate $2^{125} (\text{mod} 127)$.

Algorithms with numbers [$15 \times 4 = 60$ **points**]

11. In this exercise we shall learn to compute square roots in $\mathbb{Z}_p^*$ where p is a prime, such that $p \equiv 3(\text{mod} 4)$. We say that $x \in \mathbb{Z}_p^*$ has square root, if there exist a $y \in \mathbb{Z}_p^*$ such that $x \equiv y^2 \pmod{p}$, and y is called the square root of x.

    (a) Find the elements in $\mathbb{Z}_7^*$ which has a square root. What are their square roots?

    (b) Show that $\frac{p+1}{4}$ is an integer.

    (c) If $a \in \mathbb{Z}_p^*$ and a has a square root. Then show that $a^{\frac{p+1}{4}}$ is a square root of a.

    (d) Suppose p is a prime such that $p \equiv 3(\text{mod} 4)$. Write an algorithm which takes as input an $a \in \mathbb{Z}_p^*$ and returns the two square roots of a if the square roots exists, otherwise returns "no square roots exist". What is the running time of your algorithm.

12. Assuming that you know $\phi(N)$ for a given modulus N, devise an algorithm to compute $a^{-1} (\text{mod} N)$ using the modular exponentiation algorithm. State the running time of your algorithm.

13. Give a polynomial time algorithm for computing $a^{b^c} (\text{mod} p)$ given a, b, c and a prime p.

14. (a) Suppose we have available a black box $\mathcal{B}(.)$, which when given as input an n bit integer a returns a^2 in $O(n)$ time. Use this black-box $\mathcal{B}(.)$ to multiply two n bit numbers a and b in $O(n)$ time.

    (b) Prof. Calculus claims that there is an algorithm for squaring integers which is asymptotically faster than multiplying two integers. Argue that this claim of Prof. Calculus is false.

Programming Exercise [75 points]

15. Implement in C the Miller-Rabin test. You should take into account the following for your implementation:

    - Your implementation should take as input arbitrary long integers and give as output either PRIME or COMPOSITE depending on whether the input is a prime or a composite.
    - Implementing Miller-Rabin would require certain basic operations on integers, like modular multiplication, exponentiation, comparing two integers etc. Your program should be modular, and you should write separate functions for each of these operations so that you can re-use the code later if required.
    - You *should not* use any multi-precision libraries for this implementation.

    Important: The due date for this exercise is October 21 midnight. You should mail your code along with relevant instructions to compile and run your code to debrup.otro@gmail.com.