

ECE G205 Fundamentals of Computer Engineering
Fall 2004

Homework 2: Due by Wednesday September 22 2004

- This test contains 2 problems. They allow you to earn 100 points.
- Show your work, as partial credit can be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. **Be neat.**
- **No late submissions will be accepted.**
- Only homework returned in a 9in × 12in envelope will be accepted. (If you cannot find such envelope, ask the Instructor.) Please, write your name and the class name (ECE G205) on the envelope (write clearly, please).
- **Remote students** should send an e-mail of the (typed) solutions to the TA independently of whether code is required or not.

Write your name here: _____

- **Problem # 1 [30 points]**. Consider the *recursive* implementation of insertion sort seen in class. **(a)** Determine the worst case time complexity of that solution. **(b)** Can a function that solves the sorting problem be written that requires less than n comparisons? (n is the number of the elements in the array.) **(c)** Prove the correctness of the given implementation.

- **Problem # 2 [80 points].** Consider the problem of determining whether a given integer number $n > 1$ is prime or not. The following simple algorithm solves the problem (where the C++ operator `%` is the modulus operator, which returns the remainder of the integer division between its operands):

```
bool isPrime( int n ) {
    bool p = true;
    for ( int i = 2; i < n; i++ )
        if ( n % i == 0 )
            p = false;
    return p;
}
```

Function `isPrime` attempts to divide n by every number i in the range $2, \dots, n - 1$ and returns `true` only if no number i that divides n has been found.

- (a) Is this function time complexity polynomial in the *size of the input*? Justify your answer.
- (b) Rewrite `isPrime` by using a `while` cycle that terminates as soon as we are sure that n is not prime. When is it that this implementation has the same time complexity of the function `isPrime` above?
- (c) Evaluate function `isPrime` as if it was executed in a fast computer, that is able to execute 10^9 operations per second. How long would it take to check whether an 11 digit long number is prime? How about with numbers that are 15 and 20 digits long, respectively? The estimated age of the universe is 5×10^{17} years. What kind of number would require at least that time for checking their primality? Are these uselessly big numbers, or could they have some practical applications?
- (d) Consider the following fact: n is composite, i.e., it is not a prime number, if and only if it is divisible by $k \leq \sqrt{n}$.

Use this fact to write an algorithm for checking whether a number n is a prime number. Answer part (c) of this problem by using your new algorithm.