Homework 2
Due Thursday, June 13, 2011 by 4:00pm in class

For general homework policies and our suggestions, please see the policy document.
Sections 2.2 and 2.4 from the textbook might be useful for this homework.
For this homework, you can assume that addition, subtraction, multiplication and division of two numbers can be done in $O(1)$ time. (Side question: Is this assumption justified?)
Do not turn the first problem in.

1. (Do NOT turn this problem in) This problem is just to get you thinking about asymptotic analysis and input sizes.
An integer $n \geq 2$ is a prime, if the only divisors it has is 1 and $n$. Consider the following algorithm to check if the given number $n$ is prime or not:

For every integer $2 \leq i \leq \sqrt{n}$, check if $i$ divides $n$. If so declare $n$ to be not a prime.
If no such $i$ exists, declare $n$ to be a prime.

What is the function $f(n)$ such that the algorithm above has running time $\Theta(f(n))$? Is this a polynomial running time– justify your answer. (A tangential question: Why is the algorithm correct?)

2. (40 points)
A forest with c components is a graph that is the union of c disjoint trees. Note that a tree is a forest with 1 component. Prove that an $n$-vertex forest with $c$ components has $n - c$ edges.
(Hint: The above can be proven by induction. Of course, any other correct proof is also welcome).

Grading:
Give an example for some $c \geq 2$ showing that the property holds (you don’t have to do this part. If your proof is correct, you will receive these points. However, this can help for partial credit if your proof is incorrect): 10pts
General proof for all $n$: 30pts (40pts if you don’t give an example for some $c \geq 2$)

3. (45 points) Order the following running times in ascending order in terms of rates of growth so that if one (let’s call it $f(n)$) comes before another (let’s call it $g(n)$), then $f(n)$ is $O(g(n))$:

$2^{100n}$, $n^3$, $\log_{10}(n^4)$, $2^{\log_2 n}$, $e^{4096}$, $n! + 12^{1000}$.

Argue why your order is correct (formal proof is not needed).
(Note: It may help to review the properties of logarithms.)

Grading: This question will be graded based on how many pairs of functions are in the correct order. This means each of the 15 pairs of functions will be worth 3 points each. For each pair, the points are distributed as follows:
4. (15 points) You are commanding a military unit consisting of \( n \) spy’s. These spies all have separate missions to infiltrate \( n \) enemy bases. You are handed down orders from above your pay-grade that each spy will infiltrate the \( n \) bases in a particular order (this order can be different for each spy). It takes a spy one day to travel between any two bases and infiltrate the destination base no matter which two bases are being traveled between. Thus, each spy will be in one base each day, and multiple spies can infiltrate a single base on any given day. It is not important that each spy infiltrate each base, but they do have an alternate mission. Each spy is given one charge of C-4 powerful enough to completely destroy any one base. After a base is destroyed, no other spies can infiltrate that base and the spy that destroyed it retires and goes home. You can assume that all your spies at a base on the day it is destroyed make it out alive, but no spies can enter that base afterwards.

Your goal is to construct an algorithm to pick which base each spy will destroy such that all the bases will be destroyed. Keep in mind that once you destroy a base, no other spies can get through that base. So if a spy blows up a base on day one, and another spy is scheduled to infiltrate that base on day 2, that spy will be stuck since the base is gone. That spy will not be able to finish the rest of his mission (i.e. no bases can be skipped even if they’re destroyed).

(In real life, you will almost never come across a problem whose description will match exactly with one you will see in this course. More often, you will come across problems that you have seen earlier but are stated in a way that don’t look like the version you have seen earlier. This problem is trying to simulate that situation. In algorithms-speak, you will have to reduce the problem here to one that you have seen already.)

Correct algorithm: 5 pts
Prove the correctness of your algorithm: 10 pts