Instructions

1. **Goal of this assignment** – review of runtime analysis, recursion and Java. I strongly encourage you to submit a printed solution. Handwritten solution will only be accepted if clearly legible!

2. Review of Java includes q. 6-8. You don’t need to try make the program in question 9 work online, but it would be a good idea to compose and run some test Java programs to make sure you have the right programming environment set up. Use “java version” to make sure it’s “build 1.6.x” for some x, or at least 1.5.x, to match our default Java environment on the UNIX systems.

Questions

1. You should learn to recognize and sum a geometric series. Try these:
   
   **Note:** You don’t necessarily have to know exactly how to solve these equations. It’s enough if you look it up in your calculus text or online. The important thing for me is that you know how to look these things up and understand how they generalize.

   (a) \[ \sum_{i=1}^{10} 2^i. \]

   (b) \[ \sum_{i=1}^{\infty} (2/3)^i. \]

2. How many binary digits are there in the numbers \(2^{100}\), \(5^{100}\) and \(10^{100}\)? How are the answers to these three questions related? (**Hint:** this is a question about logarithms and change of base.)

3. (a) Show that \(\log_a(x) = c \cdot \log_b(x)\) for some constant c (expressed only in terms of the constants a and b). **Hint:** This is probably easier than you think... It follows quite directly from the log properties. You should, though, prove that it’s true for any a and b and not prove by an example.

   (b) Calculate the ratio between the number of digits required to write a number in base 10 and the number of digits required to write the same number in base 2. Notice that this question relates to question 2 and 3a above.

4. From K&T, Chapter 2, q. 1: Suppose you have algorithms with the five running times listed below. (Assume these are the exact running times.) How much slower do each of these algorithms get when you (a) double the input size, or (b) increase the input size by one?

   (a) \(n^2\)

   (b) \(n^3\)

   (c) \(100n^2\)

   (d) \(n \log n\)

   (e) \(2^n\)
5. From K&T, Chapter 2, q. 3: Take the following list of functions and arrange them in ascending order of growth rate. That is, if function \( g(n) \) immediately follows function \( f(n) \) in your list, then it should be the case that \( f(n) = O(g(n)) \).

(a) \( n^{2.5} \)
(b) \( \sqrt{2n} \)
(c) \( n + 10 \)
(d) \( 10^n \)
(e) \( 100^n \)
(f) \( n^2 \log n \)

You should find all the mathematics you need in the class notes and in Kleinberg and Tardos, chapter 2. You may find it useful to remember that one way to compare the relative growth rates of \( f(n) \) and \( g(n) \) is to look at the ratio \( f(n)/g(n) \) as \( n \to \infty \). If that ratio approaches 0, then \( g \) grows faster than \( f \): \( f(n) = O(g(n)) \). If it approaches infinity then \( f \) grows faster than \( g \). If the ratio approaches a constant different from both 0 and \( \infty \) then \( f \) and \( g \) grow at the same rate.

6. What is an interface? How does the interface differ from an abstract class? What members may be in an interface?

7. What is the difference between a final class and other classes? Why are final classes used?

8. Write Java functions (static methods) that do the following tasks. Do not compile and run, do no write an entire class, just the function itself. It should be short and simple.

(a) Given a number, return its reverse. For example – if the number is 29, return 92. Use only arithmetic operations (division and remainder come to mind), and no collections.

(b) Given a number, find if it is a palindrome. That is, if it is equal to its own reverse. For example, 101 is a palindrome, but 122 is not. (If you solved (a) it should be very simple...).

(c) Write a simple piece of code that converts a string number into its equivalent integer. For example, given the string ”1234” it should return the number 1234. Assume the string only contains the characters ’0’ – ’9’. Use character arithmetics. Do not use the parseInt or valueOf java functions.