Algorithms in the Real World (15-853), Fall 06
Assignment #7 **Due: 8 Dec 06**

You can look up material on the web and books, but you cannot look up solutions to the given problems. You can work in groups, but must write up the answers individually, and must write down your collaborators' names. Note that there are three problems on two pages. Moreover, this assignment is due ***this Friday*** *Dec 8th.*

Problem 1: Arithmetic Codes (10pt)
Given the following probability model:

Letter	$p(a_i)$	$f(a_i)$
a	.1	0
b	.2	.1
c	.7	.3

Decode the 4 letter message given by `00011011010` assuming it was coded using arithmetic coding. Why is this message longer than if we simply had used a fixed-length code of 2 bits per letter, even though the entropy of the set $\{.1, .2, .7\}$ is just a little more than 1 bit per letter. Note: once you figure out how to do the decoding, it should not take more than five minutes on a calculator or scripting language.

Problem 2: PPM, LPZ and BW (20pt)

The string `bcabccabc` is encoded using PPM. The (partial) dictionary constructed during encoding is given below. For the following questions, assume that escape count is given by the number of different characters for each context, and exclusion is *not* used, unless specified. Assume that the alphabet has 26 characters, and use $k = 2$.

(a) Fill in the empty spaces in the dictionary below.

Context	Counts	Context	Counts	Context	Counts
empty	$a = 2$	a	$b = 2$	ab	$c = 2$
	$b = 3$				
	$c = 4$	b			
		c			

Figure 1: Dictionary

(b) Suppose the next letter in the string is `b`. Compute the number of bits required to encode `b`, and also list the changes made to the dictionary. (You do not have to compute the exact number of bits, simply write it as an expression containing logs). The answer need not be an integer.

(c) Now assume that exclusion is used. Recompute the number of bits required to encode the character `b`.

(d) Encode the above string `bcabccabc` using LZ77, with an unbounded lookahead buffer, and a window of size 4.

(e) Encode the above string using Burrows-Wheeler. Just show the sequence of characters after the BW transform (don't bother compressing using move to front).

Problem 3: Estimating the Length of the Sream (10pt)

In the guest lecture on Monday Dec 4th, you saw that it was possible to estimate the number of *distinct* elements in the stream (using the Flajolet–Martin scheme), and to estimate second moment $\sum_{i \in U} f_i^2$ for the stream (using the Alon-Matias-Szegedy scheme). In this problem we consider a much simpler problem: keeping track of the number of elements in the stream. (Assume this number is at most n.)

(a) (1 point) Give a simple way to use $O(\log n)$ bits and (exactly) count the number of elements you have seen so far.

(b) (9 points) Suppose you were given only $O(\log \log n)$ bits of space. You decide to keep a counter X which you start off at 0 (i.e., $X_0 = 0$). Suppose your counter value is currently $X = X_t$, and you see another element. You set the new counter value $X_{t+1} = X_t + 1$ with probability $1/2^{X_t}$, and you maintain the value $X_{t+1} = X_t$ with the remaining $1 - 2^{-X_t}$ probability.

Here, intuitively, if 2^t elements have been seen so far, the counter X keeps track of the value $\approx t$ and hence requires only $\approx \log t$ bits. In this question we formalize this intuition.

Show that the expectation $\mathbf{E}[2^{X_N}] = N + 1$, where X_N is the random value of the counter when you have seen N elements.