CSE 490 GZ
Midterm Exam
February 8, 2002

1. Consider the probabilities on the symbols $a : 1/4, b : 3/4$. Use arithmetic coding with scaling to code the sequence *abb*. Show the steps along the way.

2. Decode the sequence 1110010 using adaptive Golomb coding with doubling.

3. Consider the following conditional probabilities for context.

	a	b	c	d
a	.25	.5	.15	.1
b	.1	.1	.1	.7
c	.3	.3	.2	.2
d	.6	.1	.1	.2

   Row x contains the probabilities of the next symbol given x is the previous symbol.

   (a) Design an Huffman tree for each context.

   (b) Encode the sequence *abcdd* using these Huffman trees. Assume a fixed prefix code of two bits per character ($a : 00; b : 01, c : 10, d : 11$) when such a code is needed.

4. Consider the following variable to fixed length code:

input	output
b	000
aa	001
ab	010
ac	011
ca	100
cb	101
cc	110

   (a) Encode the sequence *aaabbcba*. You may need to define an escape code.

   (b) If the symbols have the probabilities $a : 1/2; b : 1/4; c : 1/4$ what is the average number of bits per symbol of this code.

5. Assuming the initial indexing of the symbols $\{a, b, c, d, e\}$

0	1	2	3	4
a	b	c	d	e

use move to front to decode the following sequence: 4000400011.

6. Assuming an initial dictionary

0	a
1	b
2	c

encode the sequence aabaabb using LZW with the doubling strategy to use fewer bits when the dictionary is small.

7. Answer each of the following true or false.

   (a) Huffman coding can code arbitrarily close to the first-order entropy bound by forming treating strings of a fixed, but long enough, length as symbols.

   (b) Arithmetic coding can code arbitrarily close to the first-order entropy bound if the coded strings are long enough.

   (c) Golomb codes are are error resilient.

   (d) A lower bound on the number of bits per symbol for any compression algorithm is the first-order entropy of the data, which is based on the frequencies of the symbols.

   (e) The zero-frequency problem is the problem of how to assign codes to symbols that are seen for the first time.

   (f) The Sequitur algorithm is good for finding context-free grammars that generate the input string, but not that effective at compression.

   (g) Grade II Braille is an early compression algorithm that is no longer in use.

   (h) There is no problem in arithmetic coding in choosing the tag equal to L for the interval $[L, R)$.