Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.02 Fall 2011 **Solutions to Chapter 3** Updated on: February 8, 2012

Please send information about errors or omissions to hari; questions best asked on piazza.

1. 1 bit. 50% of the fish are bass, so the Huffman code would represent that with 1 bit (say, 0), and the other fish would all start with the bit 1.

2. (a) Yes, a valid Huffman code; it's prefix-free and corresponds, for instance, to probabilities $P(A) = 1/2, P(B) = 1/4, P(C) = 1/8, P(D) = 1/8$.

   (b) No, it isn't prefix-free. The code for B is a prefix of the code for D.

   (c) No, because a more optimal (i.e., shorter) code would have C and D encoded in 2 bits, as 10 and 11 (or vice versa), and that would be a Huffman code for the same symbol probabilities, not the one given.

3. $p_C + p_D > p_A$. The reason is that only two trees are possible. For the tree where every symbol has two bits, we need to have $p_C + p_D + p_B > p_B + p_A$, which gives us the desired answer. We need a strict inequality because otherwise there is a possibility of a tie, and the unbalanced tree may be chosed because it has the same expected number of bits.

4. The string table will have "aa", "aaa", "aaaa", "aaaaa", ... in addition to the single-byte characters (all 256 of those, including "a").

   If the receiver has received E encoded symbols, then the k^{th} of these symbols must correspond to the string "aaa...a" (k a's). Hence, the number of a's that it can decode is $E(E+1)/2$.

5. table[256] = 'ab', table[257] = 'bb', table[258] = 'bba', table[259] = 'abb', cumulative output of decoder = 'abbbabbba'.

6. (a) aabbabaa
   (b) $64 - 10 \cdot 6 = 4$ bits.
   (c) "aba"

7. (a) **True**. Whenever a new string is added to the table, the codeword corresponding to the string without the final symbol is transmitted, but the string that *gets added* may show up only much later in the text being compressed, or indeed may not show up at all! Consider, for example, the string *abbbbbbbbbbbb*... The string *ab* is added to the table early in the process, but the corresponding codeword is *never* transmitted, while the codewords corresponding to strings added later, such a *bb*, *bbb*, etc. get transmitted earlier. Thus the order in which codewords are added to the table is not necessarily related to the order in which they are transmitted.

   (b) **True.** Suppose we add the string "str + s" to the table, where "s" is a single character. At this time, we know that "str" **must** be in the table, and the encoder must transmit the codeword corresponding to "str". This argument now recursively applies to "str" as well: when we added "str" to the table, its longest-prefix (assuming "str" is of length 2

or greater) must have already been in the table, and the corresponding codeword sent. This argument applies until we get to the first character of the string "str + s". Hence, the statement is true.