We encode information so that it can be *efficiently transferred* from one place to another — whether it be between the hard drive and main memory of a computer or between people chatting on IM. This homework will explores the tradeoffs among several basic encoding schemes.

Question 1. Complete exercise 2.2.1 from *Understanding Digital Communications*.

Question 2. Complete exercise 2.2.2 from *Understanding Digital Communications*.

Question 3. In contrast to fixed length codes, *variable length codes* may encode different characters with a different number of digits. For example, consider the following variable length binary code over the American English alphabet:

A=1110	B=110000	C=01001	D=11111	E=100	F=00101	G=110011
H=0110	I=1011	J=001001011	K=0010011	L=11110	M=00111	N=1010
O=1101	P=110001	Q=001001001	R=0101	S=0111	T=000	U=01000
V=001000	W=00110	X=001001010	Y=110010	Z=001001000		

Table 1: A variable length binary code for the English Alphabet

Notice that we encode E with three bits while Q takes nine bits. This is because we expect E to appear more frequently in a message than Q. In fact, this code is optimal with respect to the frequency of letters in the American English alphabet.

(a) What is the encoding for the word *BAILOUT* using the variable length code above?

(b) Suppose we were interested in only encoding the 26 upper case letters of the American English alphabet using a fixed length binary code. What is the minimum number of binary digits per block required for such a code? How many bits are required to encode the word *BAILOUT* with such a code? Is this shorter or longer than the variable-length encoding from part (a)?

(c) When decoding a fixed-length code with block size k, one reads the first k digits and decodes the block. However, with a variable length code, no such k exists. Try decoding the phrase *00011100101110001* using the variable length code above. What does it decode to?

⋆ (d) When a scheme like the code shown in Table 1 is used to transmit messages, the binary digits are not received all at once. They may instead arrive one by one. For example, if we were receiving the message described in part (b), then at some point, we would have received the bits 111, but not know whether the next bit would be a 0 or a 1. An instant later, we would have received the bits 1110, but again, we would not know what to expect next.

The code shown in Table 1 has an important property. As soon as the last bit in a codeword arrives, we can immediately decode it. The values of the digits that arrive later do not influence the interpretation. We say that such a code is instantaneous. For example, if you receive the bits 111 while processing a message encoded using the scheme in Table 1, you cannot tell whether these bits are part of the encoding of an A, an L or a D. If the next bit is a 0, however, you can instantly tell that the entire sequence received, 1110, encodes an A. Bits received later cannot change this.

By contrast, consider the code shown below in Table 2, which is designed to encode messages containing only letters from the set {A, B, C, D}. If while receiving a message encoded using this scheme we receive the string 011111 but don't know what (if any) additional digits we will receive later, then we cannot determine whether some substring of the bits received thus far should be interpreted as CD, AD, or BD.

A=0 B=01 C=011 D=111

Table 2: A variable length binary code for A, B, C, and D

What feature of the code in Table 1 makes it instantaneous? That is, explain why the code in Table 1 is instantaneous whereas the code in Table 2 is not.

Question 4. Complete exercise 2.10.1 from *Programming with Java, Swing, and Squint*