1. (A.) Symbols carrying the most information, i.e., the symbols that are less likely to occur. This makes sense: to keep messages as short as possible, frequently occurring symbols should be encoded with fewer bits and infrequent symbols with more bits.

2. (a.) To decode the message, start at the root of the tree and consume digits as you traverse down the tree, stopping when you reach a leaf node. Repeat until all the digits have been processed. Processing the encoded message from left-to-right:

"0" => A  
"100" => B  
"0" => A  
"111" => E  
"101" => C

(b.) Using Tree #1, the expected length of the encoding for one symbol is:

\[ 1 \cdot p(A) + 3 \cdot p(B) + 3 \cdot p(C) + 3 \cdot p(D) + 3 \cdot p(E) = 2.0 \]

Using Tree #2, the expected length of the encoding for one symbol is:

\[ 2 \cdot p(A) + 2 \cdot p(B) + 2 \cdot p(C) + 3 \cdot p(D) + 3 \cdot p(E) = 2.25 \]

So using the encoding represented by Tree #1 would yield shorter messages on the average.

(c.) \( S = \{ A/0.5 \, B/0.125 \, C/0.125 \, D/0.125 \, E/0.125 \} \)

arbitrarily choose D & E
encoding: "0" => D, "1" => E

\( S = \{ A/0.5 \, B/0.125 \, C/0.125 \, DE/0.25 \} \)

choose B & C
encoding: "0" => B, "1" => C

\( S = \{ A/0.5 \, BC/0.25 \, DE/0.25 \} \)

choose BC & DE
encoding: "00" => B, "01" => C, "10" => D, "11" => E
S = \{A/0.5 BCDE/0.5\}  
choose A & BCDE  
encoding: "0" => A, "100" => B, "101" => C, "110" => D, "111" => E  

S = \{ABCDE/1.0\}  
This is Tree #1. The choice of D & E as the first symbols to combine was arbitrary---we could have chosen any two symbols from B, C, D and E. So there are many equally plausible encodings that might emerge from this algorithm, corresponding to interchanging B, C, D and E at the leaves of the tree.

3. (a.) # of bits for "C" = \log_2(1/pC) = \log_2(1/0.25) = \log_2(4) = 2  

(b.) using Huffman algorithm:
   - since D,F are least probable, make a subtree of them, p(D\&F) = 30%  
   - now A,C are least probable, make a subtree of them, P(A\&C) = 43%  
   - now B,DF are least probable, make a subtree of them P(B\&(D\&F)) = 55%  
   - just AC,BDF are left, make a subtree of them (A\&C)/(B\&(D\&F))  
   so A = 00, B = 10, C = 01, D = 110, F = 111

4 (a.) p(00)=.0668, p(01)=.4334, p(10)=.4327, p(11)=.0671  
   Huffman tree = 01 \&(10 \&(00 \& 11))  
   Code: 01="0" 10="10" 00="110" 11="111"  

   (b.) sum(p_i*len(code for p_i)) = 1.7005, which is 85\% of the original 2-bit per symbol encoding.  
   (c.) sum(p_i *log_2(1/pi)) = 1.568, so the code of 4b isn't quite optimal.

5. N = 8 choices for a 3-bit number  
   Alice: odd = \{001, 011, 101, 111\}  
   M = 4, info = \log_2(8/4) = 1 bit

   Bob: not a multiple of 3 = \{001, 010, 100, 101, 111\}  
   M = 5, info = \log_2(8/5) = .6781 bits

   Charlie: two 1's = \{011, 101, 110\}  
   M=3, info = \log_2(8/3) = 1.4150

   Deb: even, not a multiple of 3, two 1's = \{110\}
M=1, info = log₂(8/1) = 3 bits

6. (a.) p(E or O) = .31 + .19 = .5, log₂(1/p) = 1 bit

(b.) Huffman algorithm:

choose X,D: X\(\lor\)D, p = .13

choose S,XD: S\(\lor\)(X\(\lor\)D), p = .29

choose R,0: R\(\lor\)O, p = .40

choose E,SXD: E\(\lor\)(S\(\lor\)(X\(\lor\)D)), p = .6

choose RO,ESXD: (R\(\lor\)O) \(\land\) (E\(\lor\)(S\(\lor\)(X\(\lor\)D)))

code: R=00 O=01 E=10 S=110 X=1110 D=1111

(c.) 00 10 1111 110 01 1110