1. Give the ASN.1 encoding for the following three integers. Note that ASN.1 integers, like those in XDR, are 32 bits in length.

   (a) 101

INT	4	101

   (b) 10,120

INT	4	10120

   (c) 16,909,060

INT	4	16909060

2. Give the big-endian and little-endian representation for the integers from the exercise above.
   101 be 00000000 00000000 00000000 01100101
   101 le 01100101 00000000 00000000 00000000
   10120 be 00000000 00000000 00100111 10001000
   10120 le 10001000 00100111 00000000 00000000
   16909060 be 00000001 00000010 00000011 00000100
   16909060 le 00000100 00000011 00000010 00000001

3. The presentation-formatting process is sometimes regarded as an autonomous protocol layer, separated from the application. If this is so, why might including data compression in the presentation layer be a bad idea?
   It is often possible to do a better job of compressing the data if one knows something about the type of the data. This applies even to lossless compression; it is particularly true if lossy compression can be contemplated. Once encoded in a message and handed to the encoding layer, all the data looks alike, and only a generic, lossless compression algorithm can be applied.

4. Let $p \leq 1$ be the fraction of machines in a network that are big-endian; the remaining $1 - p$ fraction are little-endian. Suppose we choose two machines at random and send an `int` from one to the other. Give the average number of byte-order conversions needed for both big-endian network byte order and receiver-makes-right or both for $p = 0.1$, $p = 0.5$, and $p = 0.9$. (Hint: The probability that both endpoints are big-endian is p^2; the probability that the two endpoints use different byte orders is $2p(1 - p)$.)

   For big-endian network byte order the average number of conversions is $0 \times p^2 + 1 \times 2p(1-p) + 2 \times (1-p)^2$. For receiver-makes-right this is $0 \times p^2 + 1 \times 2p(1-p) + 0 \times (1-p)^2$; that is, if both sender and receiver are little-endian then no conversion is done. These are evaluated below:

	$p = 0.1$	$p = 0.5$	$p = 0.9$
big-endian network	1.80	1.00	0.20
receiver-makes-right	0.18	0.50	0.18

5. Suppose a file contains the letters a, b, c, and d. Nominally we require 2 bits per letter to store such a file.

   (a) Assume the letter a occurs 50% of the time, b occurs 30% of the time , and c and d each occurs 10% of the time. Give an encoding of each letter as a bit string that provides

optimal compression. (Hint: construct a Huffman code)
$a = 1, b = 01, c = 001, d = 000$

(b) What is the percentage of compression you achieve above? $1 \times 0.5 + 2 \times 0.3 + 3 \times 0.1 + 3 \times 0.1 = 1.7$
$1.7/2 \times 100 = 85\%$ as many bits, or a 15% compression gain.

(c) Repeat this, assuming a and b each occur 40% of the time, c occurs 15% of the time and d occurs 5% of the time.
The assignments are the same, although we could now give either a or b the 1-bit encoding. The average compression is now $1 \times 0.4 + 2 \times 0.4 + 3 \times 0.15 + 3 \times 0.05 = 1.8$, i.e., we use 90% as many bits or a 10% compression gain.

6. The one-dimensional discrete cosine transform is similar to the two-dimensional transform, except that we drop the second variable (j or y) and the second cosine factor. We also drop, from the inverse DCT only, the leading $1/\sqrt{2}N$ coefficient. Implement this and its inverse for $N = 8$ (spreadsheet or Matlab will do) and answer the following:

(a) If the input data is $\{1, 2, 3, 5, 5, 3, 2, 1\}$, which DCT coefficients are near 0?
For "symmetric" data such as this, coefficients *DCT (i)* for $i = 1, 3, 5, 7$ (starting the indexing at $i = 0$) should be zero or near-zero.

(b) If the data is $\{1, 2, 3, 4, 5, 6, 7, 8\}$, how many DCT coefficients must we keep so that after the inverse DCT the values are all within 1% of their original values? 10%? Assume dropped DCT coefficients are replaced with 0s.
If we keep six coefficients, the maximum error in *pixel(i)* after applying the DCT and its inverse is about 0.7%, for $i = 1$ and $i = 2$. If we keep only four or five coefficients (note that both choices lead to the same values for the inverse DCT), then the maximum error is 6%, at $i = 0$; the error at $i = 1$ is 5.6%.

(c) Let s_i, for $1 \le i \le 8$, be the input sequence consisting of a 1 in position i and 0 in position $j \ne 1$. Suppose we apply the DCT to s_i, zero the last three coefficients, and then apply the inverse DCT. Which $i, 1 \le i \le 8$, results in the smallest error in the ith place in the result? The largest error?

The input vectors for this problem look like (1, 0, 0, 0, 0, 0, 0, 0) with the 1 moving one position to the right as i increases. Here is a table listing, for each s_i, the percentage error in the ith place of the final result. The smallest error is for $i = 0$ and 7; the largest is for $i = 1$ and 6.

i	0	1	2	3	4	5	6	7
% error	12.3	53.1	39.6	45.0	45.0	39.6	53.1	12.3

7. Compare the size of an all-white image in JPEG format with a typical photographic image of the same dimensions. At what stage or stages of the JPEG compression process does the white image become smaller than the photographic image?
The all-white image generates all zeros in the DCT phase. The quantization phase leaves the 8×8 grid of 0s unchanged; the encoding phase then compresses it to almost nothing.

8. Suppose you want to implement fast-forward and reverse for MPEG streams. What problems do you run into if you limit your mechanism to displaying I frames only? If you don't, then to display a given frame in the fast-forward sequence, what is the largest number of frames in the original sequence you may have to decode?

If you display I frames only, then the fast-forward speed is limited to the rate at which I-frames are included; note that this may be variable.

The worst case for decoding an arbitrary frame is when the frame you want is a B frame. It depends on a previous P frame P and a future P or I frame Q. To decode the B frame you

want, you will first need P and its I frame, and also Q. If Q is a P frame, then its I frame is the same as that of P. The total number of frames processed, including the one wanted, is four.