1) TRUE
2) FALSE
3) FALSE
4) TRUE
5) TRUE
6) C
7) A
8)
9)
10) FALSE
11) TRUE
12) FALSE
13) FALSE
14) TRUE
15) B
16) B
17) B
18) A
19) D
20) A
21) B
22) B
23) byte
24) nibble
25) 256
26) Hexadecimal
27) TRUE
28) FALSE
29) TRUE
30) TRUE
31) TRUE
32) TRUE
33) FALSE
34) TRUE
35) FALSE
36) A
37) A
38) B
39) B
40) D
41) Emoticons
42) netiquette
43) shareware
44) fair use
45) Phishing
46) open source
47) TRUE
48) FALSE
49) TRUE
50) TRUE

51) TRUE
52) FALSE
53) TRUE
54) TRUE
55) D
56) B
57) C
58) D
59) Decryption
60) XOR
61)
62)

61) Recall that `C` in hexadecimal is 12 in decimal. Thus, to convert `2C` from hexadecimal to decimal, we multiply 2×16^1 and add it to 12×16^0, which gives $32 + 12 = 44$. In general, if we have an $n + 1$ digit hexadecimal number $x_n x_{n-1} x_{n-2} \cdots x_1 x_0$, where each x_i is a hexadecimal digit, we can convert the number into decimal by computing

$$(x_n \times 16^n) + (x_{n-1} \times 16^{n-1}) + (x_{n-2} 16^{n-2}) + \cdots + (x_1 \times 16^1) + (x_0 \times 16^0).$$

62) To convert 13 from decimal to binary, we use the following steps:

1. We first determine what is the largest power of 2 that is no more than 13. Note that $2^4 = 16$ and $2^3 = 8$, so 8 is the largest power of 2 no more than 13. Now 8 goes into 13 once with a remainder of 5, so our first bit is 1.
2. Then we work with the next smaller power of 2, which is $2^2 = 4$. Note that 4 goes into the current remainder 5 once with a remainder of 1, so our next bit is 1.
3. Then we work with the next smaller power of 2, which is $2^1 = 2$. Note that 2 goes into the current remainder 1 zero times with a remainder of 1, so our next bit is 0.
4. Then we work with the next smaller power of 2, which is $2^0 = 1$. Note that 1 goes into the current remainder 1 one time with a remainder of 0, so our next bit is 1.
5. There are no more smaller nonnegative powers of 2, so we are done. We read off the bits from the above steps in order to arrive at the final answer: 1101.

63)

plaintext	`N`	`J`	`I`	`T`
ASCII binary	`01001110`	`01001010`	`01001001`	`01010100`
key	`00010111`	`00101101`	`00010111`	`00101101`
Encrypted binary	`01011001`	`01100111`	`01011110`	`01111001`
Encrypted ASCII	`Y`	`g`	`^`	`y`