Hexadecimal Representations

1. Convert the following 32-bit binary patterns to octal and hexadecimal representation. (See note: *Bits and Bit Patterns*)

   (a) 01110010 11110101 00111101 00001001 16275236411, 72 F5 3D 09

   (b) 10100010 11011111 11101001 00111000 24267764470, A2 DF E9 38

   (c) 01101110 10001111 10101101 01010010 15643726522, 6E 8F AD 52

   (d) 01011001 11001110 00110111 10001101 13163433615, 59 CE 37 8D

   (e) 10110001 11011001 11110101 01100100 26166372544, B1 D9 F5 64

   (f) 01011101 10001110 00101111 10100011 13543427643, 5D 8E 2F A3

Character Representations

2. Convert the following bits to ASCII (8 bit) characters. (See note: *Representing Characters with Bit Patterns*)

   (a) 01001000 01100101 01101100 01101100 01101111 00100001 Hello!

   (b) 01000011 01001101 01010011 00110010 00110011 00110000 CMS230

   (c) 01000110 01100001 01101100 01101100 00110001 00111000 Fall18

   (d) 01010010 01101111 00100001 00100001 01101001 01101110 01110011 Ro!!ins

   (e) 01000011 00110000 01101100 01101100 01100101 01100111 01100101 C0llege

   (f) 01010100 00110100 01110010 01110011 T4rs

3. Convert the following strings to a sequence of bytes. (Recall that in C, strings are really just an array of characters, terminated by the null character.) Give your answer in hex notation. (See note: *Representing Characters with Bit Patterns*)

   (a) `"Ab12"` 41 98 31 32 00

   (b) `"4& hH"` 34 26 20 68 48 00

   (c) `"y%Z6"` 79 25 5A 36 00

   (d) `"U(9)"` 55 28 39 29 00

   (e) `"$_mY"` 24 5F 6D 59 00

   (f) `"@Gm8"` 40 47 6D 38 00

Other Base Systems

4. Convert the following quantities to their base-10 (decimal) representations. (See note: *Number Representation*)

   (a) 102_3 11

   (b) 41_5 21

   (c) 62_8 50

   (d) 515_6 191

   (e) 111_4 21

   (f) 614_7 305

5. Convert the following decimal values to the specified base system. (See note: *Binary and Hex Integer Representation*)

   (a) 342 to base 3 110200_3

   (b) 189 to base 4 2331_4

   (c) 1229 to base 5 14404_5

   (d) 5023 to base 6 35131_6

   (e) 4782 to base 7 16641_7

   (f) 7612 to base 9 11387_9

Unsigned Integer Representation

6. Convert the following 8-bit patterns to positive decimal numbers. (See note: *Number Representation*)

   (a) 00001001 9 (d) 10000101 133

   (b) 00111000 56 (e) 01100100 100

   (c) 01010010 82 (f) 11001111 207

7. Convert the following positive decimal numbers to 8-bit binary numbers. (See note: *Binary and Hex Integer Representation*)

   (a) 18 00010010 (d) 108 01101100

   (b) 25 00011001 (e) 243 11110011

   (c) 63 00111111 (f) 175 10101111

Signed Integer Representation

8. Convert the following negative decimal numbers to 8-bit 2's complement binary numbers. (See note: *Binary Addition and Two's Complement*)

   (a) -29 11100011

   (b) -86 10101010

   (c) -63 11000001

   (d) -15 11110001

   (e) -105 10010111

   (f) -71 10111001

9. Convert the following 8-bit patterns to signed decimal numbers. Assume these patterns use 2's complement representation. (See note: *Binary Addition and Two's Complement*)

   (a) 10001011 -117

   (b) 10111000 -72

   (c) 10101011 -85

   (d) 10011101 -99

   (e) 10001111 -113

   (f) 10010011 -109

10. Perform the following 2's complement additions. Give the end result (in binary format) and state whether or not an overflow occurs. (See note: *Binary Addition and Two's Complement*)

    (a)
    ```
     0111 0011
     1101 0010
    -----------
    ```
    Solution: 0100 0101, no overflow

    (b)
    ```
     1111 0111
     1001 1010
    -----------
    ```
    Solution: 10010001, no overflow

    (c)
    ```
     1111 0111
     0101 1010
    -----------
    ```
    Solution: 01010001, no overflow

    (d)
    ```
     1101 0111
     1001 1010
    -----------
    ```
    Solution: 01110001, overflow

    (e)
    ```
     1101 0111
     1001 1010
    -----------
    ```
    Solution: 01110001, overflow

    (f)
    ```
     0101 0111
     0011 1010
    -----------
    ```
    Solution: 10010001, overflow

Floating Point Representation

11. Convert the following decimal values to IEEE-754 single precision (32-bit) floating point format. (See note: *Floating Point Representation*)

    (a) -2.875 11000000 00111000 00000000 00000000

    (b) 3.375 01000000 01011000 00000000 00000000

    (c) 6.75 01000000 11011000 00000000 00000000

    (d) 27.0 01000001 11011000 00000000 00000000

    (e) -12.5 11000001 01001000 00000000 00000000

    (f) -14.875 11000001 01101110 00000000 00000000

12. Convert the following patterns to a decimal value, assuming they are IEEE-754 single precision (32-bit) floating point format. (See note: *Floating Point Representation*)

    (a) 01000001 10101100 00000000 00000000 21.5

    (b) 01000000 11101100 00000000 00000000 7.375

    (c) 01000001 01111100 00000000 00000000 15.75

    (d) 01000001 01100000 00000000 00000000 14.0

    (e) 11000001 11111100 00000000 00000000 -31.5

    (f) 11000001 11101110 00000000 00000000 -29.75