CIS 2107 Lab 6 – Float Stuff

The purpose of this lab is to practice manipulating 32-bit floating point numbers. You will write code to extract both the bit value and, numeric values of a float. The code required is:

1. A typedef struct flt for the sign, exponent, mantissa and mode of a float.
2. A main function to test all your functions after completion.
3. A function get_flt_bits_int to get the bits of a float as an int for bitwise manipulation.
4. A function get_flt_sign_char to get the char value of a float
5. A function get_flt_sign_val to get the numeric sign of a float.
6. A function get_flt_exp_str to return a string of the bits in the exponent of a float.
7. A function get_flt_exp_val to return the integer value of the exponent in a float with respect to the bias.
8. A function get_flt_exp_mode to get the mode of a float from the exponent.
9. A function get_flt_man_str to get a string containing the bit value of the mantissa in a float.
10. A function get_flt_man_val to get the float value of the mantissa in a float.
11. A function get_flt_bits_str to return a string with the sign, exponent and mantissa of a float with each part separated by a space.
12. A function get_flt_val_flt that converts a float to a flt struct.
13. A function that prints a flt structs data to screen.
14. A function get_flt_bits_val that converts a flt struct back to a float.

Download the floatStuff.c file from Canvas. Read the comments about the content of the code in each function. There are lots of hints to help you complete this project. I strongly suggest that you write the functions in order as they appear above because subsequent functions depend on previous functions.

Write a function and text it before moving to the next. I included the preprocessor statements to help. Note that the #defines can all be interpreted as integer values in code. The main is at the bottom to avoid typing prototypes after the preprocessor statements. Good luck!

Sample screen output for sqrt(-1):

```
f = -1.#IND00

sig = 1
s = -1

exp = 11111111
e = 255

man = 10000000000000000000000
m = 0.5000000000

bits = 1 11111111 10000000000000000000000

sign = -1
```

```
exp = 255
man = 0.500000
mode = specialized

ff = 1.#QNAN0


Process returned 0 (0x0)     execution time : 5.604 s
Press any key to continue.
```

Sample output for INFINITY

```
f = 1.#INF00

sig = 0
s = 1

exp = 11111111
e = 255

man = 00000000000000000000000
m = 0.0000000000

bits = 0 11111111 00000000000000000000000

sign = 1
exp = 255
man = 0.000000
mode = specialized

ff = 1.#INF00


Process returned 0 (0x0)     execution time : 0.976 s
Press any key to continue.
```

Sample screen output for -15.375

```
f = -15.375000

sig = 1
s = -1

exp = 10000010
e = 3

man = 11101100000000000000000
m = 0.9218750000

bits = 1 10000010 11101100000000000000000

sign = -1
```

```
exp = 3
man = 0.921875
mode = normalized

ff = -15.375000


Process returned 0 (0x0)    execution time : 1.075 s
Press any key to continue.
```