COMP 222 Computer Organization
Assignment #2—Decimal and IEEE-754 Conversion

Objective:

To convert a number between Decimal and IEEE-754 single-precision floating-point standard format.

Inputs:

- Number in Decimal format (including special case of 0)
- Number in IEEE-754 format (including special cases: +0, -0, +∞, -∞, NaN, denormalized)

Output:

- Equivalent number in IEEE-754 single-precision floating-point standard format
- Equivalent number in Decimal

Specification:

The program converts a number based on choosing from a menu of choices, where each choice calls the appropriate procedure, where the choices are:

1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Quit program

- Make sure your code compiles with zyBooks' zyLabs compiler--if it does not compile with zyBooks' compiler it will be graded as not compiling, even if it compiles on your compiler on your desktop/laptop at home
- To mask a number to extract a field, use "variable" & "constant" (i.e. var & 0x80000000 to extract the first "bit" of var)
- To represent a constant in hexadecimal, use "0x" before the hexadecimal constant
- To print a number in hexadecimal format, use "%x"
- To use the math library, use "include <math.h>" for functions like "pow(x,y)" to raise "x" to the "y" power
- Feel free to use the template "skeleton" code provided on Canvas for the assignment

What NOT to do (any violation will result in an automatic score of 0 on the assignment):

- Do NOT modify the choice values (1, 2, 3) or input characters and then try to convert them to integers--the test script used for grading your assignment will not work correctly.
- Do NOT turn in an outdated version of the assignment downloaded from the Internet (coursehero, github, etc.) or a version that was coded by someone else (former student, tutor, etc.)
- Do NOT use any self-created or external libraries that cannot be located/utilized by zylabs
- Do NOT turn in your assignment coded in another programming language (C++, C#, Java, Python, Perl, etc.)—it will NOT compile under zyLabs C compiler.

What to turn in:

The source code as a single C file uploaded to Canvas (http://canvas.csun.edu) by the deadline of 11:59pm PST (-20% per consecutive day for late submissions, up to the 4th day—note 1 minute late counts as a day late, 1 day and 1 minute late counts as 2 days late, etc.).

Sample test run (Inputs: 1 2.5 2 40200000 1 0 2 80400000 2 ffffffff 3)—Note: inputs in the test run may not show up on the output display

```
Floating-point conversion:
--------------------------
1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Exit

Enter selection: 1
Enter the decimal representation: 2.5
Sign: 0
Biased exponent: 10000000
Mantissa: 01000000000000000000000
IEEE-754 format: 40200000

Floating-point conversion:
--------------------------
1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Exit

Enter selection: 2
Enter the IEEE-754 representation: 40200000
Sign: +
Unbiased exponent: 1
Normalized decimal: 1.250000
Decimal format: 2.500000

Floating-point conversion:
--------------------------
1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Exit

Enter selection: 1
Enter the decimal representation: 0
Sign: 0
Biased exponent: 00000000
Mantissa: 00000000000000000000000
IEEE-754 format: 00000000

Floating-point conversion:
--------------------------
1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Exit

Enter selection: 2
Enter the IEEE-754 representation: 80400000
Sign: -
Unbaised exponent: -126
Denormalized decimal format: -0.500000*2^(-126)

Floating-point conversion:
--------------------------
1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Exit

Enter selection: 2
Enter the IEEE-754 representation: ffffffff
Special case: NaN

Floating-point conversion:
--------------------------
1) Decimal to IEEE-754 conversion
2) IEEE-754 to Decimal conversion
3) Exit

Enter selection: 3
```