Project 1
Data Type Converter

The purpose of this project is to become more familiar with bit-level representations of integers and floating-point numbers.

In this project, you will implement an application, in C or Java programming language; that takes

a file containing signed integers, unsigned integers and floating-point numbers the byte ordering type (Little Endian or Big Endian),
the size of the floating-point data type (the size of both signed and unsigned integers are 2 bytes),

as input and converts the contents of the file to hexadecimal according to the predefined format and gives the converted data as output.

The size of the data type can be 1, 2, 3, 4, or 6 bytes.

The signed integers are written as normal integers in the file. (e.g., 3, -5, etc.)
The unsigned integers are written as integers followed by the letter u. (e.g., 3u, 5u, etc.)
The floating-point numbers are written with a decimal point. (e.g., 3.0, 5.987, etc.)
If the read data type is signed integer, your program will convert the numbers in the input file using 2's complement representation.
If the read data type is unsigned integer, numbers will be converted using unsigned integer representation.
If the read type is floating point number, you will use IEEE-like format. The number of exponent bits according to given data size will be like the following:

- if 1 byte (i.e., 8 bits), 4 bits will be used for exponent part
- if 2 bytes (i.e., 16 bits), 6 bits will be used for exponent part
- if 3 bytes (i.e., 24 bits), 8 bits will be used for exponent part
- if 4 bytes (i.e., 32 bits), 10 bits will be used for exponent part
- While calculating the mantissa to get the floating-point value, you will only use the first 13 bits of the fraction part (for 3-byte and 4-byte data sizes). You will use "round to nearest even" method for rounding fraction bits to 13 bits.

At the beginning of the execution, your program will prompt for the input file.

In the input file, each line will include a single number.

Example: input.txt

29.109375
4u
-9
6u
0
-63.0

After a valid input file is taken as input, the user will be prompted for the byte ordering type and size:

Example:

Byte ordering: Little Endian
Floating point size: 2 bytes

You program will calculate the hexadecimal value of the file content with the given information:

Example:

The byte ordering is Little Endian, and the floating point is 2 bytes.

- First, your program will read the first number of the file. For our "input.txt" the first number is:
  29.109375
- This is a floating-point number.
  29.109375 = 11101.000111 = 1.1101000111 * 2^4
- Mantissa is **1.1101000111**. However, for 2 bytes, we have 9 bits of fraction part, so we need to round to nearest even: **1.1101000111 ≈ 1.110100100**. The fraction part is **110100100**
- E = 4. Exponent is E = exp – Bias, where Bias = 2^{6-1} – 1 = 31. Therefore, the equation is 4 = exp - 31. exp = 35 which is **100011** in binary.
- The number is positive, so sign bit is 0.
- The number at total is **0100011110100100** which is **0x47A4** in hexadecimal.
- The byte ordering is Little Endian, so the floating-point number is represented as:
  A4 47

- If the program reads the following number, it is:
  4u
- This is an unsigned integer. Unsigned integers are 2 bytes:
  4 = 0000000000000100
- **0000000000000100** is **0x0004** in hexadecimal.
- The byte ordering is Little Endian, so the unsigned integer is represented as:
  04 00
  In the output file, the printed value will be: **04 00**

The output file will be:

output.txt

A4 47
4 00
F7 FF
6 00
0 00
F0 C9