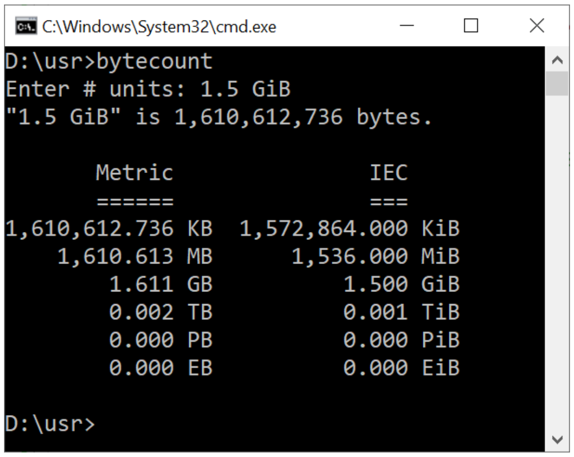
Project #1 – Do you know your numbers?

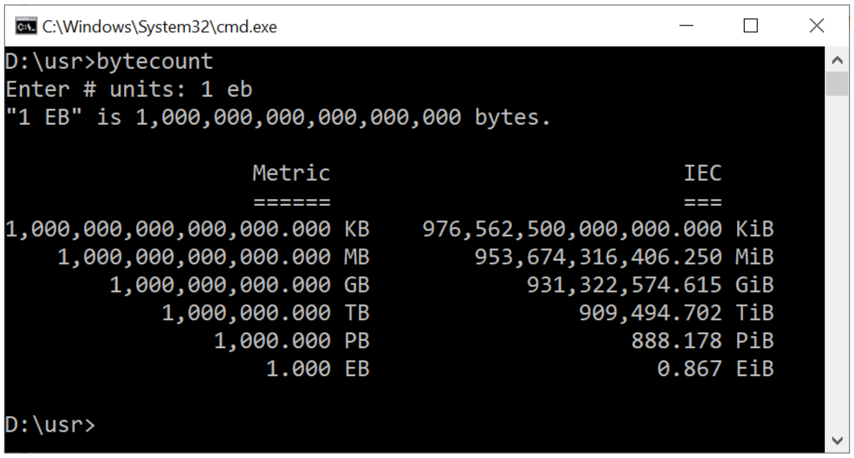
Course	INFO-1156 Object-Oriented Programming in C++
Professor	Garth Santor, and Lynn Koudsi
Assigned	Tuesday, May 4th, 2021
Due	Friday, May 28th, 2021 by 11:59 pm
Weight	6%

Project Description (v1.0.0)

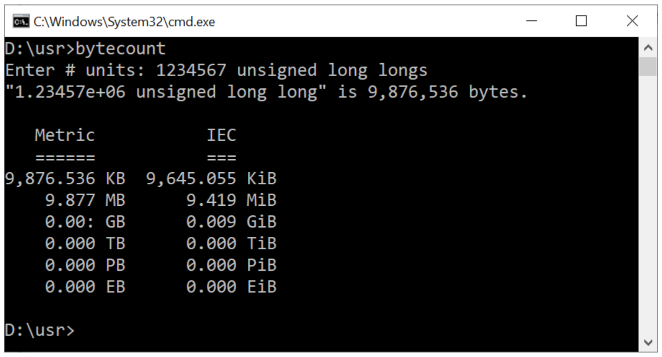
Write a C17 (**not** C++) console application that converts storage units to its equivalents.



```
D:\usr>bytecount
Enter # units: 1.5 GiB
"1.5 GiB" is 1,610,612,736 bytes.

      Metric               IEC
      ======               ===
1,610,612.736 KB  1,572,864.000 KiB
    1,610.613 MB      1,536.000 MiB
        1.611 GB          1.500 GiB
        0.002 TB          0.001 TiB
        0.000 PB          0.000 PiB
        0.000 EB          0.000 EiB

D:\usr>
```



```
D:\usr>bytecount
Enter # units: 1 eb
"1 EB" is 1,000,000,000,000,000,000 bytes.

                 Metric                             IEC
                 ======                             ===
1,000,000,000,000,000.000 KB    976,562,500,000,000.000 KiB
    1,000,000,000,000.000 MB        953,674,316,406.250 MiB
        1,000,000,000.000 GB            931,322,574.615 GiB
            1,000,000.000 TB                909,494.702 TiB
                1,000.000 PB                    888.178 PiB
                    1.000 EB                      0.867 EiB

D:\usr>
```



```
D:\usr>bytecount
Enter # units: 1234567 unsigned long longs
"1.23457e+06 unsigned long long" is 9,876,536 bytes.

   Metric          IEC
   ======          ===
9,876.536 KB  9,645.055 KiB
    9.877 MB      9.419 MiB
    0.00: GB      0.009 GiB
    0.000 TB      0.000 TiB
    0.000 PB      0.000 PiB
    0.000 EB      0.000 EiB

D:\usr>
```

Your program should prompt the user to enter a number followed by a unit type. The program then prints out the number of bytes represented, followed by a table of the equivalent multi-byte units.

Only one input line!

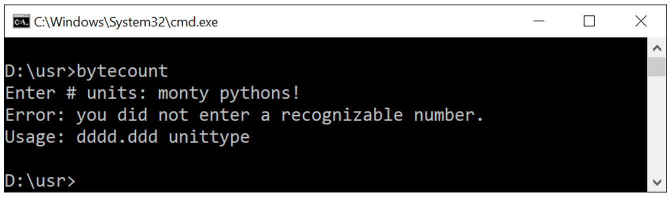
Unit types are B, kB, MB, GB, TB, PB, EB, KiB, MiB, GiB, TiB, PiB, EiB[1], char, unsigned char, short, unsigned short, int, unsigned int, long, unsigned long, long long, unsigned long long, float, and double.

[1] We are not supporting zettabytes, zebibytes, yottabytes, or yobibytes as they would exceed the capacity of an **unsigned long long**.

Unit information can be found at https://en.wikipedia.org/wiki/Kilobyte.

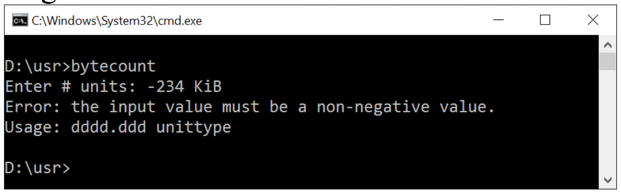
Error Examples:

Non-numeric value.



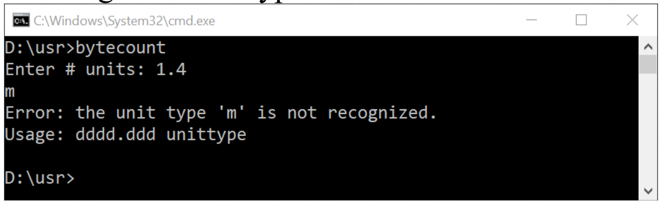
```
C:\Windows\System32\cmd.exe

D:\usr>bytecount
Enter # units: monty pythons!
Error: you did not enter a recognizable number.
Usage: dddd.ddd unittype

D:\usr>
```

Negative value.

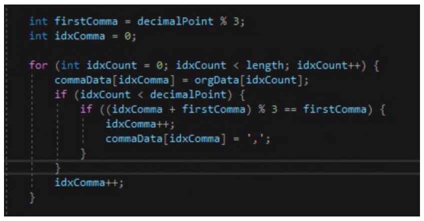


```
C:\Windows\System32\cmd.exe

D:\usr>bytecount
Enter # units: -234 KiB
Error: the input value must be a non-negative value.
Usage: dddd.ddd unittype

D:\usr>
```

Unrecognized unit type.



```
C:\Windows\System32\cmd.exe
D:\usr>bytecount
Enter # units: 1.4
m
Error: the unit type 'm' is not recognized.
Usage: dddd.ddd unittype

D:\usr>
```



```
int firstComma = decimalPoint % 3;
int idxComma = 0;

for (int idxCount = 0; idxCount < length; idxCount++) {
    commaData[idxComma] = orgData[idxCount];
    if (idxCount < decimalPoint) {
        if ((idxComma + firstComma) % 3 == firstComma) {
            idxComma++;
            commaData[idxComma] = ',';
        }
    }
    idxComma++;
}
```

You must code your solution with the following restrictions:

- The source code, **must be C**, not C++.
- Must compile in Microsoft Visual C with /std:c17
- Output messages should match the order and content of the demo program precisely.
- The executable file must be named 'bytecount.exe'
- Do not worry about input overflow (numbers that are too large), that is an issue beyond the scope of this project.

Design

Follow the input-compute-output pattern.

Inputs

1. the amount of the specified units
2. the type of the specified units

Computation

We will use a concept from mathematics called normalization, where the input value is normalized to a common unit, then the output is generated from the normalized unit. For example, a conversion from inches to metres would *normalize* inches to feet (the basic unit of imperial measurement), then convert from feet to metres. 33 inches → $\frac{33}{12} = 2.75$ feet → $2.75 \times 0.3048 = 0.8382$ metres.

What would be the *normalized* unit be for memory amounts? Bytes!

Output

A table of SI units (metric) and their equivalent IEC units (binary).

Development

Don't code straight to the final program – work your way there in steps. Here is a development plan you can try:

1. Read a real number and a single word. Output that number and that word. **Test to see that it accepts fractions (e.g., 3.5, 42.1, 3.0, etc.)**
2. Output the correct number of bytes for that input number and single-word type. **Test every conversion – with and without fractions**. What are your test values? Pick boundary values like 1 KB, 1 KiB, 1.99 MiB, 2.5 EiB, etc.
3. Output the table of conversions. **Test again with your same values**.
4. Align the output columns.
5. Make the input case insensitive (input can be upper or lower case, or any mix).
6. Make the output display the correct capitalization (e.g., 'KiB', not 'kib').
7. Allow units to be plurals. (e.g., KBs)
8. Check for non-numeric input (report, then quit if found).
9. Check for negative input (report, then quit if found).
10. Check for no matching unit type (e.g., '10 kg' should produce an error, then quit).
11. Handle the multiple word types.
12. Handles any amount of whitespace in between the words.
    Test examples: **"10TB",**
    "10 unsigned char",
    "   10   signed   short   int   "
13. Numbers printed with thousand separators. There is no standard library function for this so you'll have to write your own.
14. Handles the 'EiB' problem. C processes numbers followed by an 'e' as scientific notation (e.g., "8e2" is accepted as 8×10^2 or 800). So "2EiB" would be mistaken as a badly-formed scientific notation whereas "2 EiB" is handled correctly. Fix this so that "2EiB" is correctly parsed as "2 Exbibytes".

Grading Criteria

Difficulty:	Normal	Moderate	Difficult	Elite

NOTICE! All of the features of one difficulty level must be completed before we mark the next group (i.e., we will not mark yellow features, if the green features are incomplete.

Criteria	Weight	Max Score	Score	Value
Functional Requirements				
Accepts a 'real' number for input	3%	1	1	3%
Input validation: input number displayed to 3 decimal places	2%	1	1	2%
Accepts single word unit types	5%	1	1	5%
Outputs the correct number of bytes (within rounding error is 4/5)	25%	5	5	25%
Outputs the table of conversions (in columns)	5%	1	1	5%
Columns are aligned	10%	1	1	10%
Columns adjust to longest number size	5%	1	1	5%
Input is case insensitive	5%	1	1	5%
Output has the correct case (e.g., 'kib' becomes 'KiB', etc.)	5%	1	1	5%
Plurals are ignored	5%	1	1	5%
Reports non-numeric input and quits.	5%	1	1	5%
Reports negative input value and quits.	5%	1	1	5%
Reports no matching unit type	5%	1	1	5%
Handles multi-word types (between the number and the newline)	5%	1	1	5%
Handles any number of whitespaces in between the words (not \n)	5%	1	1	5%

Prints with thousand separators (your code)	3%	1	1	3%
Handles the eib problem (e.g., '234eib' confuses scanf)	2%	1	1	2%
Non-functional requirements				
Visual Studio project doesn't generate a program named 'bytecount.exe'	-10%		0	0%
Penalties from C & C++ Grading Guide v2.2.0	-5%		0	0%
Late submission (days)	-10%		0	0%
Total	100%			**100%**

Submission Requirements

1. Submit **entire Visual Studio project directory** to Fanshawe Online
   a. Delete ***all*** debug and release directories.[i]
   b. Submit in a .ZIP, .7z archive file.

[i] Alternatively, you can 'clean' your project for submission by downloading 'vsclean' a Visual Studio Solution Cleaner from https://www.gats.ca/software/vsclean/.