MAT 4370: Programming Assignment 1

Due: Friday, January 18

Limits for Integer Types

The `<limits.h>` header provides macros which allow the C programmer to determine the range of each integer type. Implement a program, named `displayLimits.c`, which uses these macros to display a compact summary of the ranges for signed and unsigned varieties of `char`, `short`, `int`, and `long` types.

Your summary should include ranges for each of the following:

- `char` — signed, unsigned, and unmodified
- `short` — signed and unsigned
- `int` — signed and unsigned
- `long` — signed and unsigned

The first few lines of output from your program should appear as follows:

```
char
----
   signed char range: -128 ... 127
   unsigned char range: 0 ... 255
   char range: -128 ... 127

short
-----
   short int range: -32768 ... 32767
   unsigned short int range: 0 ... 65535
```

Avoid "hard-coded" constants — use the appropriate macros in `<limits.h>`.

Displaying Integer Values

Suppose a `long` variable holds the number of bytes in a certain file. We wish to display this value in one of two ways:

- As one sometimes sees numbers expressed in everyday life, with commas showing grouped digits; for example, 12,345,678 rather than 12345678.
- As seen in the Unix `ls` command using the `-lh` "human-style" sizes, showing the number of bytes (B), kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T). In this context, the prefixes kilo, mega, giga and tera refer to the IEEE 1541 binary standards ($2^{10}, 2^{20}, 2^{30}$, and 2^{40}) and not their usual scientific meaning. For sizes which require a suffix other than B, the value should be given to the nearest tenth. For example, a file with 10,136 bytes should be displayed as 9.9K and a file with 123 bytes should be displayed as 123B.

Design and implement a C program, named `displayLong.c`, which will prompt for and read one `long` value from the keyboard and display its value in each of the two ways described above. Your program should include two functions to produce the desired output.

Coding Practices

My expectation for the programs you write are as follows:

- All programs should include appropriate comments. At a bare minimum, include your name, the assignment number, the date completed, a brief explanation of what it does, and any known bugs. Additionally, include running commentary which helps a reader understand how the program achieves its goal.
- Use correct spelling in all of your comments.
- Use meaningful variable names.
- Avoid use of "magic numbers." Use `#define` to introduce constants instead.
- Use appropriate amounts of "white space" — both vertical and horizontal — to make it easier to perceive the program text.
- Assume the program text will be printed on a standard sized sheet of paper. Do not use overly long lines which would extend past the usual margins. Typically, this means ensuring lines do not exceed 80 characters.
- Ensure that your program compiles cleanly. Use the `-Wall` compiler switch and pay close attention to any warnings given.
- Thoroughly test your programs. If you are aware of deficiencies, document these as "bugs."

What to Submit

When your programs compile cleanly and work to your satisfaction, place them in a folder named `hw01`. Drag this folder onto the SUBMIT icon, found in the Applications folder.

Tips for Success

- Start early, allowing time for mistakes and surprises.
- Plan ahead. It is often best to do your thinking away from the computer. Programming done "off the top of your head" in front of a computer is likely to be prone to errors and will lead to excessive and frustrating debugging sessions.
- Don't underestimate the effort: even things which look easy may have subtle aspects.
- Take pride in your work: a well-designed and implemented computer program can be just as rewarding as other creative activities.
- Carefully digest the assigned reading.

About Due Dates

Assignments for this class are due **at the beginning of class** on the stated due date.