CSCI 2500 — Computer Organization
Lab 02 (document version 1.0) — Due 10 February 2021
Everything is just ones and zeros

- This lab is due by the Midnight EST Wednesday via a Submitty gradeable.
- This lab is to be completed **individually**. Do not share your code with anyone else.
- Labs are available on Mondays before your lab session. Plan to start each lab early and ask questions during office hours, in the Submitty forum, and during your lab session.

1. **Checkpoint 1:** The concept of **endianness** plays an important role on your laptop (or any computer hardware architecture). Specifically, endianness dictates whether the most significant byte (i.e., "big-end") or least significant byte (i.e., "little-end") is stored as the first byte of the value being stored at a given address. Some architectures are big-endian (e.g., Motorola 68K, older IBM processors), while most other architectures are little-endian (e.g., x86 and its descendants). See below for more info:

   https://en.wikipedia.org/wiki/Endianness

   Start with copying the code below. Attempt to understand what's happening in the given code as you copy it in to your editor. The key to understanding this code is that an `int` data type in C is stored as four bytes while characters are a single byte.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  /* Insert your four bytes of ASCII for your secret message */
  /* The 0x prefix above indicates a hexadecimal number */
  int z = 0x4F4F4F57;

  char* c = (char*)&z;

  printf("%c", *c++);
  printf("%c", *c++);
  printf("%c", *c++);
  printf("%c\n", *c);

  return 0;
}
```

   Compile and run this code, then replace the hexadecimal value of variable `z` with your own secret four-letter word. To do so, look up each letter in an ASCII table and convert them to their corresponding hexadecimal values. You can find an ASCII table at the following URL:

   https://en.wikipedia.org/wiki/ASCII#ASCII_control_code_chart

   Based on your output, what is the endianness of your system? You'll need to know this for the subsequent checkpoints. Hint: its probably little-endian.

2. **Checkpoint 2:** Download the `lab02-data.[your-endianness].dat` file and attempt to decipher what the file contains. From the shell, you can try viewing the file in a text editor or use the `cat` and `hexdump` to learn what you can about the file. Any guesses as to what data is stored in this file?

   Assume that the data file contains an array of `int` variables. Write a C program to read this data file entirely into memory, then display the data using a loop. You'll need to open the file, determine its size in bytes, allocate an array, and then read the binary data into the array. You can easily accomplish this using the following functions:

   `calloc()`, `fopen()`, `fseek()`, `ftell()`, `fread()`, and `fclose()`.

   Note: `fread()` reads in binary data directly to some memory address. `fseek()` and `ftell()` can get you the size of the file.

   Finally, display the data as follows (hint: "%3d" might be useful):

   ```
   Data point #  0: <int-value>
   Data point #  1: <int-value>
   Data point #  2: <int-value>
   ...
   Data point #183: <int-value>
   Data point #184: <int-value>
   Data point #185: <int-value>
   ```

3. **Checkpoint 3:** Modify your solution for Checkpoint 2 by instead assuming that the given data file contains an array of `unsigned long` variables.

   Display the data as follows:

   ```
   Data point #  0: <unsigned-long-value>
   Data point #  1: <unsigned-long-value>
   Data point #  2: <unsigned-long-value>
   ...
   Data point # 90: <unsigned-long-value>
   Data point # 91: <unsigned-long-value>
   Data point # 92: <unsigned-long-value>
   ```

   Did you figure out what this data file contains yet?

   Note that we can interpret the raw data however we like, as its really just a bunch of ones and zeros. We can treat it as an array of `int` values, an array of `unsigned long` values, a string of printable characters, etc. But, only one interpretation turns out to be useful. Such is life.

4. **Submission Details:** Fill in the provided template code given in `lab02.c`. This has empty functions for Checkpoints 2 and 3. You'll submit this code to Submitty. The expected output format is given as above. You don't need to submit anything for Checkpoint 1.