Fall '18 CIS 314 Assignment 1 – 100/100 points – Due Wednesday, 10/3, 11:59 PM

Please submit individual source files for coding exercises (see naming conventions below). Your code and answers need to be documented to the point that the graders can understand your thought process. Full credit will not be awarded if sufficient work is not shown.

You'll also need to ensure that your C code can be compiled and run correctly using *gcc*. We'll be covering installation and setup of VirtualBox and Linux virtual machines in labs this week. Note that macOS comes bundled with *clang*, which is aliased as the *gcc* command – these are not strictly equivalent.

Please review the course plagiarism policy (see the syllabus) prior to completing the assignment. Any sources consulted other than the textbook or course slides must be cited, and no code may be copy and pasted from another student, past or present. Failure to follow either of these guidelines may result in immediate failure of the course.

1. [25] Consider the following C code:

```
#include <stdio.h>

void printBytes(unsigned char *start, int len) {
    for (int i = 0; i < len; ++i) {
        printf(" %.2x", start[i]);
   }
    printf("\n");
}

void printInt(int x) {
    printBytes((unsigned char *) &x, sizeof(int));
}

void printFloat(float x) {
    printBytes((unsigned char *) &x, sizeof(float));
}
```

Copy and paste the above code into a file named 1-1. Add the following functions to the file, taking the indicated data type as a parameter and calling *printBytes* as appropriate: *printShort*, *printLong*, *printDouble*.

Also write a *main()* function to test each of the above functions (with the exception of *printBytes*, which you do not need to test directly) with reasonable inputs. Do you notice anything unexpected regarding the output of your test functions? List any observations as comments inline in your main function and explain these observations based on your readings from the textbook.

2. [25] Suppose we number the bytes in a 32-bit word from 0 (least significant) to 3 (most significant). Write code for the following C function that will return an unsigned int consisting of bytes 3 and 2 from *x* and bytes 1 and 0 from *y*:

unsigned int combine (unsigned int x, unsigned int y);

Here are some test runs:

```
combine(0x12345678, 0xABCDEF00): 0x1234EF00
combine(0xABCDEF00, 0x12345678): 0xABCD5678
```

Use only bitwise operators; no if statements, loops, or arithmetic operators (+, -, *, /, %). Also write a main() function to test your function.

Hint: use a bit mask to isolate bytes 3 and 2 from *x*, and another bit mask to isolate bytes 1 and 0 from *y*, then combine the results.

Name your source file 1-2.c

3. [25] Suppose we again number the bytes in a 32-bit word from 0 (least significant) to 3 (most significant). Write code for the following C function that will return an unsigned int such that byte *i* of *x* is replaced by byte *b*:

unsigned int replace (unsigned int x, int i, unsigned char b);

Here are some test runs:

```
replace(0x12345678, 2, 0xAB): 0x12AB5678
replace(0x12345678, 0, 0xAB): 0x123456AB
```

Use only bitwise operators; no if statements, loops, or arithmetic operators (+, -, *, /, %). Also write a main() function to test your function.

Hint: use shifts to align the input byte and a mask to isolate the bytes that should remain.

Name your source file 1-3.c

4. [25] Suppose we number the bits in a 32-bit word from 0 (least significant) to 31 (most significant). Write code for the following C function that will return 1 if *x* has at least one bit with a value of 1 at an odd index, 0 otherwise:

int oddBit(unsigned int x);

Here are some test runs:

```
oddBit(0x1): 0
oddBit(0x2): 1
oddBit(0x3): 1
oddBit(0x4): 0
oddBit(0xFFFFFFFF): 1
oddBit(0xAAAAAAAA): 1
oddBit(0x55555555): 0
```

Use only bitwise and logical operators (&&, ||, !); no if statements, loops, arithmetic operators (+, -, *, /, %), or conditional operators (==, !=, etc). Also write a main() function to test your function.

Hint: use a bit mask to isolate the odd bits, and think about the effect of a logical *not* (or two) on an int in C.

Name your source file 1-4.c

Zip the source files and solution document (if applicable), name the .zip file <Your Full Name>Assignment1.zip (e.g., EricWillsAssignment1.zip), and upload the .zip file to Canvas (see Assignments section for submission link).