Fall '17 CIS 314 Assignment 1 – 100/100 points – Due Monday, 10/2, 11:59 PM

Please submit individual source files for coding exercises (see naming conventions below) and a single solution document for non-coding exercises (.txt or .pdf only). 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.

1. [25] B&O'H 2.57.

Write procedures show_short, show_long, and show_double that print the byte representations of C objects of types short int, long int, and double, respectively. Try these out on several machines.

See Figure 2.4 for an example. Also write a main() function to test your procedures. Name your source file 2-57.c

2. [25] B&O'H 2.60.

Suppose we number the bytes in a w-bit word from 0 (least significant) to w/8 − 1 (most significant). Write code for the following C function, which will return an unsigned value in which byte i of argument x has been replaced by byte b:

```
unsigned replace_byte (unsigned x, int i, unsigned char b);
```

Here are some examples showing how the function should work:

```
replace_byte(0x12345678, 2, 0xAB) --> 0x12AB5678
replace_byte(0x12345678, 0, 0xAB) --> 0x123456AB
```

Use only bitwise operators; no loops or conditionals (e.g., *for*, *if* statements). Also write a main() function to test your function. Name your source file 2-60.c

3. [25] B&O'H 2.64.

Write code to implement the following function:

```
/* Return 1 when any odd bit of x equals 1; 0 otherwise.
   Assume w=32. */
int any_odd_one(unsigned x);
```

Your function should follow the bit-level integer coding rules (page 120).

Assume that the most-significant, third-most-significant bit, etc. are odd and that the least-significant bit is even.

Here are some test runs:

0x0: 0
0x1: 0
0x2: 1
0x3: 1
0xFFFFFFFF: 1
0x55555555: 0

Also write a main() function to test your function. Name your source file 2-64.c

4. [25] B&O'H 2.68.

Write code for a function with the following prototype:

```
/*
 * Mask with least signficant n bits set to 1
 * Examples: n = 6 --> 0x3F, n = 17 --> 0x1FFFF
 * Assume 1 <= n <= w
 */
int lower_one_mask(int n);
```

Your function should follow the bit-level integer coding rules (page 120). Be careful of the case n = w.

Hint: The reason that you need to be careful of the case in which n = w is because the effect of shifting by w is undefined in standard C. Think about a solution involving shifting by (n – 1) instead.

Here are some test runs:

1: 1
2: 3
3: 7
5: 31

Also write a main() function to test your function. Name your source file 2-68.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).