

Fall '18 CIS 314 Assignment 2 – 100/100 points – Due Wednesday, 10/10, 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. [20] 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 a bit mask containing 1s for the least-significant n bits and 0s for the remaining most-significant bits:

```
int mask(int n);
```

Your solution will need to handle values of n on the range $[0, 31]$ (i.e., 0 up to, and including, 31).

Here are some test runs:

```
mask(0) : 0x0
mask(1) : 0x1
mask(2) : 0x3
mask(3) : 0x7
mask(5) : 0x1F
mask(8) : 0xFF
mask(16) : 0xFFFF
mask(31) : 0x7FFFFFFF
```

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

Hint: you can use subtraction on this one – please do.

Name your source file 2-1.c

2. [20] Suppose we number the bytes in a 32-bit word from 0 (least significant) to 3 (most significant) and that the word consists of 4 individual signed bytes. Write code for the following C function that will return byte i of x sign extended to 32 bits:

```
unsigned int extract (unsigned int x, int i);
```

Here are some test runs:

```
extract(0x12345678, 0) : 0x00000078
extract(0xABCDEF00, 2) : 0xFFFFFCD
```

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

Hint: arithmetic shift is your friend with regard to sign extension.

Name your source file 2-2.c

3. [15] Fill in the missing expression in the following C code such that it will return 1 if x is >= y, 0 otherwise (you can assume that neither argument is NaN and that +0 and -0 are considered equal):

```
int ge(float x, float y) {
    unsigned int ux = *((unsigned int*) &x); // convert x raw bits
    unsigned int uy = *((unsigned int*) &y); // convert y raw bits
    unsigned int sx = ux >> 31; // extract sign bit of ux
    unsigned int sy = uy >> 31; // extract sign bit of uy
    ux <=< 1; // drop sign bit of ux
    uy <=< 1; // drop sign bit of uy
    // TODO: return using sx, sy, ux, uy
}
```

Here are some test runs:

```
ge(0.0f, 0.0f): 1
ge(-0.0f, 0.0f): 1
ge(-1.0f, 0.0f): 0
ge(0.0f, 1.0f): 0
ge(1.0f, 0.0f): 1
ge(0.0f, -1.0f): 1
```

Use only bitwise and relational (==, !=, <, <=, >, >=) operators; no if statements, loops, or arithmetic operators (+, -, *, /, %). Also write a main() function to test your function. Name your source file 2-3.c

Hint: you're going to want to "or" together all of the cases in which the function will return 1. For example, (ux == 0 && uy == 0) will be true only if x is either +0.0f or -0.0f and y is either +0.0f or -0.0f.

4. [15] Convert the following hex values to decimal assuming that they are stored as 32-bit 2s complement integers.

a. (5) 0x000000C6

b. (5) 0xFFFFFEC6

c. (5) 0xFFFFFFFF

Write your answers in your solutions document. Show your work.

5. [15] Convert the following hex values to decimal assuming that they are encoded as IEEE 754 single-precision floating-point numbers:

a. (5) 0x80000000

b. (5) 0x439D0000

c. (5) 0xC1340000

Write your answers in your solutions document. Show your work.

6. [15] Convert the following decimal numbers to hex encoded as IEEE 754 single-precision floating-point numbers. Write your answers in your solutions document.

a. (5) -1.0

b. (5) 16.75

c. (5) -314.0

Write your answers in your solutions document. Show your work.

Zip the source files and solution document (if applicable), name the .zip file <Your Full Name>Assignment2.zip (e.g., EricWillsAssignment2.zip), and upload the .zip file to Canvas (see Assignments section for submission link).