Name: \_\_\_\_\_Grace Miguel\_\_\_\_ Date: \_\_\_2/26/21\_\_\_

Pledge: \_\_\_\_\_I pledge my honor that I’ve abided by the Stevens Honor System.\_\_\_\_\_

For each function below, trace through it with reasonably small integer values. What does each function do?

**Requirement:** You should assume integers are only **8 bits** for the purpose of this exercise. The sign bit is the leftmost of the 8 bits.

**int** **mystery1**(**int** a, **int** b) {

**int** c = a - b,

d = (c >> 7) & 1,

mystery = a - c \* d;

**return** mystery;

}

Trace: mystery1(3, 7) returns \_\_7\_\_

C = -4

D= 1

Mystery = 3-(-4\*1) = 7

Trace: mystery1(8, 7) returns \_\_8\_\_

C = 1

D = 0

Mystery = 8-(1\*0) = 8

Summary: This function bitshifts c to the right by 7, and checks to find the higher number.

**int mystery2**(**int** x) {

**return** (x && !(x & (x - 1)));

}

Trace: mystery2(1) returns \_\_\_1\_\_\_

Trace: mystery2(2) returns \_\_1\_\_\_

Trace: mystery2(3) returns \_\_0\_\_

Trace: mystery2(4) returns \_\_0\_\_

Trace: mystery2(5) returns \_\_1\_\_\_

Trace: mystery2(6) returns \_\_0\_\_\_

Trace: mystery2(7) returns \_\_0\_\_\_

Trace: mystery2(8) returns \_\_1\_\_\_

Summary: Checks to see if x & x-1 = 1. If that is true, than it returns the opposite and checks that against x. This works for 2^x.

**int** **mystery3**(**int** x, **int** y) {

**int** s, c;

s = x ^ y;

c = x & y;

**while** (c != 0) {

c = c << 1;

x = s;

y = c;

s = x ^ y;

c = x & y;

}

**return** s;

}

Trace: mystery3(5, 7) returns \_12\_\_

S=2

C=5

---

C=10

X=2

Y=10

C=12

S=8

C=2

--

C=4

X=7

Y=4

S=12

C=0

Trace: mystery3(2, 8) returns \_10\_\_\_

S=10

C=0

Return 10

Summary: \_returns the sum of the two inputs