2024-2 SP-Assignment-1 Problems

1. Suppose we are given the task of generating code to multiply integer variable x by various different constant factors K. To be efficient, we want to use only the operators +, -, and <<. For the following values of K, write C expressions to perform the multiplication using at most three operations per expression.

A.
$$K = 5$$
B. $K = 9$
C. $K = 32$
D. $K = -56$
 $-64+8$
 $(\chi(2)+\chi)$
 $(\chi(3)+\chi)$
 $\chi(5)$
 $-(\chi(6)+(\chi(3))$

2. Write C expressions to generate the bit patterns that follow, where aⁿ represents n repetitions of symbol a. Assume a w-bit data type. Your code may contain references to parameters m and n, but not a parameter representing w.

A.
$$0^{w-n}1^n$$
 (($< M$)-1) $< M$

3. We are running programs on a machine where values of type **int** are 32 bits. They are represented in 2's-complement, and they are right shifted arithmetically. Values of type **unsigned** are also 32 bits. We generate arbitrary values of x and y, and converted them to **unsigned** values as follows:

For each of the following C expressions, you are to indicate whether or not the expression always yields 1. If it always yields 1, describe the underlying mathematical principles. Otherwise, give an example of arguments that make it yield 0.

A.
$$(x > y) == (-x < -y)$$
 false. $Z = \{ y = INT_{MIN}, -X = -1 - y = INT_{MIN} \}$

B. $((x + y) << 5) + x - y == 31 * y + 33 * x + true$

C. $-x + -y == (x + y) + 1$ false $-X = xx + 1 - y = xy + 1$

D. $(int)(ux - uy) == -(y - x)$

E. $((x >> 1) << 1) <= x$

Thue $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

Thuse $(x >> 1) << 1$ for $x > x$

T

4. Fill in the return value for the following procedure, which tests whether its first argument is greater then or equal to its second. Assume the function f2u returns an unsigned 32-bit number having the same bit representation as its floating-point argument. You can assume that neither argument is NaN. The two flavors of zero, +0 and -0, are considered equal.

/* Give an expression using only ux, uy, sx, and sy */

return
$$((\zeta X = -\zeta Y) ? (UX \ge UY) \cdot (\zeta X < \zeta Y)$$

5. Given a floating-point format with a k-bit exponent and an n-bit fraction, write formulas for the exponent E, significand M, the fraction f, and the value V for the quantities that follow. In k exp, n frac addition, describe the bit representation of exponent and fraction.

A. The number 6.0
$$E^{-2}$$
 M^{-2} f^{-2} V^{-10} V^{-6}

B. The largest odd integer that can be represented exactly E=M M=1.11...1 f=11...1 $V=2^{M-1}$

JE.M. F.V

C. The reciprocal of the smallest positive normalized value

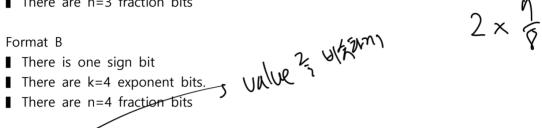
$$2^{2+1}$$
 $E = -2^{k-1} + 2 \text{ M} = 1 \text{ } f = 0 \text{ } V = 2^{-2^{k+2}} = 2^{k}$

6. Intel-compatible processors also support an "extended precision" floating-point format with an 80-bit word divided into a sign bit, k=15 exponent bits, a single integer bit, and n=63 fraction bits. The integer bit is an explicit copy of the implied bit in the IEEE floating-point representation. That is, it equals 1 for normalized values and 0 for denormalized values. Fill in the following table giving the approximate values of some "interesting" numbers in this format.

714-2= 16982	
Description	Value
Largest positive denormalized	$2^{-16382} \times (1-2^{-63})$
Smallest positive normalized	2-16382
Smallest negative normalized	-> (6782

- 7. Consider the following two 9-bit floating-point representations based on the IEEE floating-point format. Z125 , 3
 - 1. Format A
 - There is one sign bit
 - There are k=5 exponent bits.
 - There are n=3 fraction bits

2. Format B



Below, you are given some bit patterns in Format A, and your task is to convert them to the closest value in Format B. If rounding is necessary, you should round toward +∞. In addition, give the values of numbers given by the Format A and Format B bit patterns. Give these as whole numbers (eg. 17) or as fractions (eg. 17/64 or 17/26).

ገና >

5 m = 15 Glas = 7 Format B Format A **Bits** Value Value Bits -1.125 X 2-1 હ -0.5625 10110 PD10 1 01110 001 = - 0.5625 2×11 = 112 0 10101 (11)0 1100 101 112 1526 = -0.00883446UA 000000000 1 00111 110 5/x1-80/2 0.00870077(21-0 00000 111 -28 x 36 -496.800 -212_ 4096 1111 /111 1 (11011) 000 28 XIN 784 784 Q (/// /000 0 10111 100 23

312 125

8921 ()(1 8. We are running programs on a machine where values of type **int** have a 32-bit 2's-complement representation. Values of type **float** use the 32-bit IEEE format, and values of **double** use the 64-bit IEEE format. We generate arbitrary integer values x, y, and z, and convert them to values of type **double** as follows:

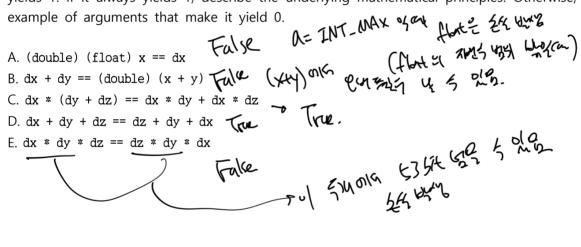
```
/* Create some arbitrary values */
int x = random();

int y = random();

int z = random();

/* Convert to double */
double dx = (double) x;
double dy = (double) y;
double dz = (double) z;
```

For each of the following expressions, you are to indicate whether or not the expression always yields 1. If it always yields 1, describe the underlying mathematical principles. Otherwise, give an example of arguments that make it yield 0.



#