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 4+1 (x<<2)+x
   B. K = 9 8+1 (x<<3)+x
   C. K = 32 32 x<<5
   D. K = -56 -64+8 -(x<<6)+(x<<3)

2. Write C expressions to generate the bit patterns that follow, where a^n 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. $0^{w-n}1^n$ (1<<n)-1
   B. $0^{w-n-m}1^n0^m$ ((1<<n)-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:

```
/* Create some arbitrary values */
int x = random();
int y = random();
/* Convert to unsigned */
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
```

-1 1
-2 INT_MIN
1 -1
2 INT_MIN

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. x=1 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= ~x+1. -y= ~y+1
D. `(int)(ux - uy) == -(y - x)` true, 비트 구조는 다르지 않음
E. `((x >> 1) << 1) <= x` true << 하면 빈칸을 0으로 채우기 때문!

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.

$x \geq y$?

```
int float_ge(float x, float y) {
    unsigned ux = f2u(x);
    unsigned uy = f2u(y);
    /* Get the sign bits */
    unsigned sx = ux >> 31;
    unsigned sy = uy >> 31;

    /* Give an expression using only ux, uy, sx, and sy */
    return (____________________);
}
```

(sx==sy) ? (ux ≥ uy) : (sx < sy)

(

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 addition, describe the bit representation of exponent and fraction.

k exp, n frac

→ E. M f. V

A. The number 6.0 2^2+2 $E=2$ $M=\frac{6}{4}=1.1_2$ $f=\frac{2}{4}=0.1_2$ $V=110_2=6$

B. The largest odd integer that can be represented exactly $E=n$ $M=1.11\ldots1_2$ $f=11\ldots1_2$ $V=2^{n+1}-1$

C. The reciprocal of the smallest positive normalized value

2^2+1

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

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.

$2^{14}-2=16382$

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

80 bit (정규화 1)

1 15 63

7. Consider the following two 9-bit floating-point representations based on the IEEE floating-point format.

bias = 3 2^2

1. Format A
   - There is one sign bit
   - There are k=5 exponent bits.
   - There are n=3 fraction bits

$2 \times \frac{7}{8}$

2. Format B
   - There is one sign bit
   - There are k=4 exponent bits.
   - There are n=4 fraction bits → value 끼리 비교하기

Below, you are given some bit patterns in Format A, and your task is to convert them to the $\boxed{\text{closest value}}$ 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 $17/2^6$).

1 5 3 1 4 4

bias = 15 Format A		bias = 7 Format B	
Bits	Value	Bits	Value
1 01110 001 (14)	$-1.125 \times 2^{-1} = -0.5625$	1 0110 0010	-0.5625
0 10101 110 (21, 3/4, 15+6)	$2^6 \times 1.75 = 112$	0 1101 1100	112
1 00111 110 (7)	$-2^{-8} \times 1.75 = -0.0068359375$	0 0000 0000	0
0 00000 111	$2^{-15} \times 1.875 = 0.000057220459$	0 0000 0001	0.0083007812
1 11011 000 (27)	$-2^{12} = -4096$	1 1111 1111	$-2^8 \times \frac{31}{16} = -496.875$
0 10111 100 (23)	$2^8 \times 1.5$ 384	0 1111 1000	384

875
125/7

$\frac{31}{16} \times \frac{1}{256}$

127 19

87 15

8421
1111

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;
```

→ 32bit 정수

float 32

double 64

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.

A. `(double) (float) x == dx` False a= INT_MAX 일때 float은 손실 발생 (float의 자릿수 범위 벗어날 수)

B. `dx + dy == (double) (x + y)` False (x+y)에서 오버플로우 날 수 있음.

C. `dx * (dy + dz) == dx * dy + dx * dz` → True.

D. `dx + dy + dz == dz + dy + dx` True

E. `dx * dy * dz == dz * dy * dx`

False

→ 이 두개에서 53bit 넘을 수 있음 손실 발생

#