CSED211 컴퓨터SW시스템개론

20180551
컴퓨터공학과
이준석

1. <2.60>

[답안]

```
unsigned replace_byte(unsigned x, int i, unsigned char b)
{
        int mask;
        unsigned res = x;
        mask = 0xff;

        mask = mask << (i * 8);
        mask = ~mask;

        res = res & mask;
        res = res | b << (i * 8);

        return res;
}
```

2. <2.68>

[답안]

```
int lower_one_mask(int n)
{
        int t;
        t = 1 << 31;
        t = t >> (31 - n);
        t = ~t;
        return t;
}
```

3. <2.73>

[문제]

Write code for a function with the following prototype:
/* Addition that saturates to TMin or TMax */
int saturating_add(int x, int y);
Instead of overflowing the way normal two's-complement addition does, saturating addition returns *TMax when there would be positive overflow, and TMin when there would be negative overflow. Saturating arithmetic is commonly used in programs that perform digital signal processing.*
Your function should follow the bit-level integer coding rules (page 164).

[답안]

```
int saturating_add(int x, int y)
{
        int sum = x + y;
        int w = (sizeof(int) << 3) - 1;
        int mask = (~(x ^ y) & (y ^ sum) & (x ^ sum)) >> w;
        int TmaxTmin = (1 << w) ^ (sum >> w);
        return  ((~mask) & sum) + (mask & TmaxTmin);
}
```

4. <2.83>

[문제]

Consider numbers having a binary representation consisting of an infinite string of the form 0.*y y y y y y . . . ,* *where y is a k-bit sequence. For example, the binary*
representation of 1
3 is 0.*01010101 . . . (y* = 01), while the representation of 1
5 is
0.*001100110011 . . . (y* = 0011).
A. Let *Y = B2Uk(y), that is, the number having binary representation y. Give a formula in terms of Y and k for the value represented by the infinite string. Hint: Consider the effect of shifting the binary point k positions to the right.*
B. What is the numeric value of the string for the following values of y?
(a) 101
(b) 0110
(c) 010011

[답안]

(A)

k칸 만큼 소수점을 오른쪽으로 옮긴 것(Y * 2^k)을 만들고 그것에 원래의 수(Y)를 빼면 y가 나온다.

그것을 수식으로 표현하면
y / (2^k - 1) = Y
Y와 k에 관해서 표현하면,
y = Y * (2^k - 1)

(B)

위의 식에 대입하여 계산하면 된다.
(a) 5/(2^3-1) = 5/7
(b) 6/(2^4-1) = 2/5
(c) 19/(2^6-1) = 19/63

5. <2.88>

[문제]

Consider the following two 9−bit floating−point representations based on the IEEE floating−point format.
1. Format A
There is 1 sign bit.
There are k = 5 exponent bits. The exponent bias is 15.
There are n = 3 fraction bits.
2. Format B
There is 1 sign bit.
There are k = 4 exponent bits. The exponent bias is 7.
There are n = 4 fraction bits.
In the following table, 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* $+\infty$. In addition, give the values of numbers given by the format A and format B bit patterns. Give these as whole numbers (e.g., 17) or as fractions (e.g., 17*/64 or 17/2*6).

[답안]

1 01111 001	-9/8	1 0111 0010	-9/8
0 10110 011	176	0 1110 0110	176

1 00111 010	-5/1024	1 0000 0101	-21/2048
0 00000 111	7/131072	0 0000 0001	17/2048
1 11100 000	-8192	1 1110 1111	-248
0 10111 100	384	0 1111 0000	+infinity (256)

6. <2.94>

[문제]

Following the bit-level floating-point coding rules, implement the function with the following prototype:
/* Compute 2*f. If f is NaN, then return f. */
float_bits float_twice(float_bits f);
For floating-point number *f , this function computes 2.0 . f . If f is NaN, your function should simply return f .*
Test your function by evaluating it for all 2^{32} values of argument f and comparing the result to what would be obtained using your machine's floating-point operations.

[답안]

```
float_bits float_absval(float_bits f)
{
        int exp = (f >> 23) & 0xFF;
        // +- zero와 denormalized value, frac 비트 부분을 왼쪽으로 한 칸 이동시키면 된다.
        if (exp == 0)
        {
                return (f & 0x80000000) | (f << 1);
        }
        // NaN, 그대로 반환한다.
        if (exp == 0xFF)
```

```
        {
                return f;
        }
        // normalized value는 exp 비트 부분에 1을 더해주면 된다.
        f = f + (1 << 23);
        return f;
}
```

7.

[문제]

For a single precision floating point number, it uses 32 bits. In fact, there are 2 ^ 32 different presentations possible using 32 bits. However, some presentations are not floating point number representation (ex. NaN). Find how many presentations are meaningful floating point number representation?

[답안]

2^32가지에서 NaN과 +infinity와 -infinity를 제외한 것들을 표현할 수 있다.

즉, exp 비트 부분이 전부 1인 경우의 수를 제외하면 된다.

2 * 2^23 = 2^24

2^32-2^24 = 2^24*(2^8-1) = **(2^24 * 255) 가지**의 수를 표현할 수 있다.

(plus zero와 minus zero를 다른 것으로 본 경우이다.

같은 것으로 본다면 1개를 빼주어야 한다.)