作业——1

0. 请证明补码加法公式：$[x]_补+[y]_补=[x+y]_补$（mod 2^w）。 $[*]_补$表示整型数据*的补码表示，机器字长为 w。

1. 将 8 位无符号数 130 转换为 8 位浮点数（exp 域宽度为 4 bits, frac 域宽度为 3bits）
Exp = ?
Frac = ?

2. We are running programs on a machine with the following characteristics:

- Values of type int are 32 bits. They are represented in two's complement （补码）, and they are right shifted arithmetically. Values of type unsigned are 32 bits.
- Values of type float are represented using the 32-bit IEEE floating point format, while values of type double use the 64-bit IEEE floating point format.

We generate arbitrary values x, y, and z, and convert them to other forms as follows:

```
/* Create some arbitrary values */
int x = random();
int y = random();
int z = random();
/* Convert to other forms */
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
double dx = (double) x;
double dy = (double) y;
double dz = (double) z;
```

For each of the following C expressions, you are to indicate whether or not the expression always yields 1.

Expression	Always True?
`(x<y) == (-x>-y)`	Y N
`((x+y)<<4) + y-x == 17*y+15*x`	Y N
`~x+~y+1 == ~(x+y)`	Y N
`ux-uy == -(y-x)`	Y N
`(x >= 0) || (x < ux)`	Y N
`((x >> 1) << 1) <= x`	Y N
`(double)(float) x == (double) x`	Y N
`dx + dy == (double) (y+x)`	Y N
`dx + dy + dz == dz + dy + dx`	Y N

3. In the following questions assume the variables **a and b are signed integers** and that the machine uses two's complement representation. Also assume that MAX_INT is the maximum integer, MIN_INT is the minimum integer, and W is one less than the word length (e.g., W = 31 for 32-bit integers). Match each of the descriptions on the left with a line of code on the right (write in the letter).

//1's Complement：反码，即按位取反

//2's Complement：补码

1. One's complement of a ________
2. a. ________
3. `a & b.` ________
4. `a * 7.` ________
5. `a / 4 .` ________
6. `(a < 0) ? 1 : -1 .` ________

a. `~(~a | (b ^ (MIN_INT + MAX_INT)))`

b. `((a ^ b) & ~b) | (~(a ^ b) & b)`

c. `1 + (a << 3) + ~a`

d. `(a << 4) + (a << 2) + (a << 1)`

e. `((a < 0) ? (a + 3) : a) >> 2`

f. `a ^ (MIN_INT + MAX_INT)`

g. `~((a | (~a + 1)) >> W) & 1`

h. `~((a >> W) << 1)`

i. `a >> 2`

4. Match each of the assembler routines on the left with the equivalent C function on the right.

```
foo1:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     sall $4,%eax
     subl 8(%ebp),%eax
     movl %ebp,%esp
     popl %ebp
     ret

foo2:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     testl %eax,%eax
     jge .L4
     addl $15,%eax
.L4:
     sarl $4,%eax
     movl %ebp,%esp
     popl %ebp
     ret

foo3:
     pushl %ebp
     movl %esp,%ebp
     movl 8(%ebp),%eax
     shrl $31,%eax
     movl %ebp,%esp
     popl %ebp
     ret
```

```
int choice1(int x)
{
    return (x < 0);
}

int choice2(int x)
{
    return (x << 31) & 1;
}

int choice3(int x)
{
    return 15 * x;
}

int choice4(int x)
{
    return (x + 15) /4
}

int choice5(int x)
{
    return x / 16;
}

int choice6(int x)
{
    return (x >> 31);
}
```