**2.58：**

Int is\_little\_endian(){

Int x = 1; //最低字节为1

Return \*((char\*)&x); //取低址字节，小端模式为1大端模式为0

}

**2.61:**

A. !~x //所有位为1

B. !x //所有位为0

C. !~(x | ~0xff) //最低有效字节所有位为1

D. !( (x>> ((sizeof(int)-1)<<3)) & 0xff) //最高有效字节所有位为0

**2.77:**

A. K=17: (x<<4) + x //2^4+1=17

B. K=-7: x-(x<<3) //1-2^3=-7

C. K=60: (x<<6)-(x<<2) //2^6-2^2=60

D. K=-112: (x<<4)-(x<<7) //2^4-2^7=-112

**2.84:**

((ux<<1)==0 && (uy<<1)==0)|| //同为0，符合题意

(sx && !sy)|| //x为负，y为正，符合题意

(!sx && !sy && ux <= uy)|| //同为正数，绝对值大的大

(sx && sy && ux >= uy) //同为负数，绝对值大的小

**2.89:**

A. ture, 先强制转换double并不影响再强制转换float

B. false, 当x-y越界时,左边double不会越界，而右边int会越界。

C. true,double可以精确表示所有正负2^53以内的所有整数。所以三个数相加可以精确表示

D. false, double无法精确表示2^64以内所有的数,该表达式很有可能不会相等。反例可以考虑比较大的值。

E. false, 0/0.0 为NaN, (非0)/0.0 为正负inf。同号inf相减为NaN,异号inf相减也为被减数的inf。故该表达式可能不会相等。比如0/0.0!=1/1.0

**2.91:**

A.pi的二进制数表示为: 0 10000000 10010010000111111101011, E=128-127=1,它表示的二进制小数值为: 11 .001001000011111101011

B.根据2.83，可知1/7的表示为0.001001[0011....

所以22/7为11.001001001001001[0011...

C.从第9位开始不同。