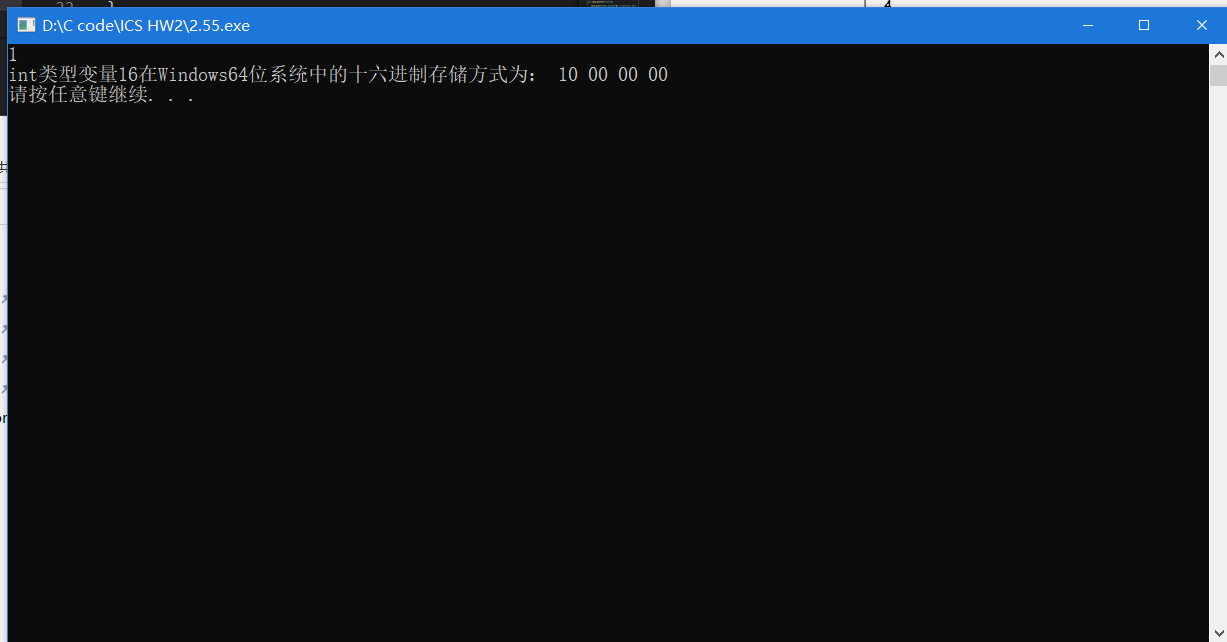
## 2.55



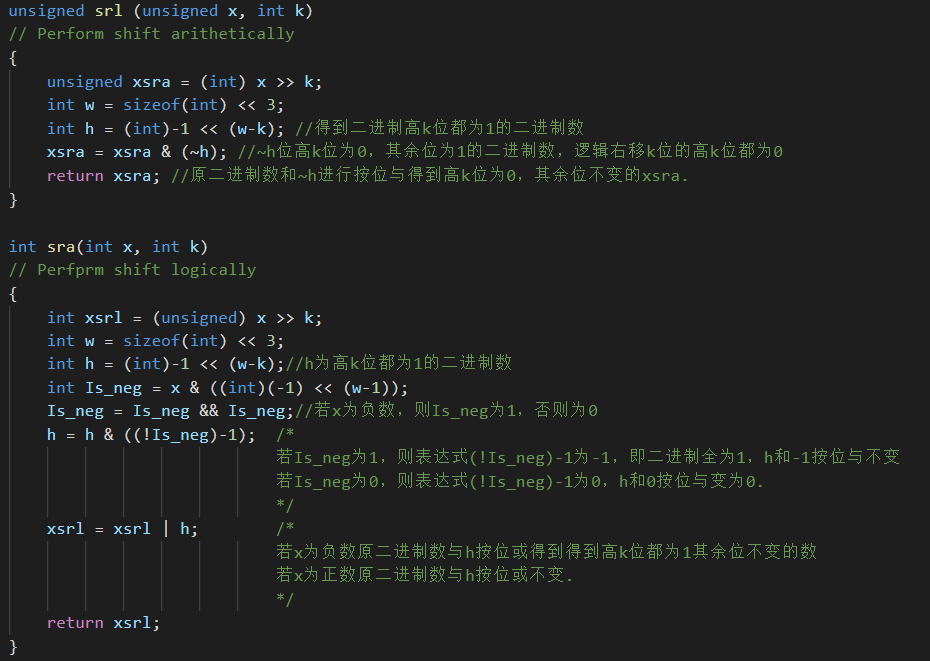


由上图可知，我是用的Win与Linux系统下数据都是小端法存储的。

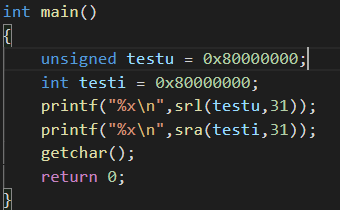
## 2.59

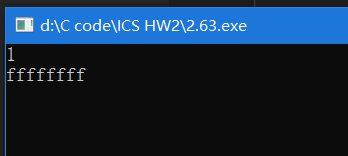


## 2.63

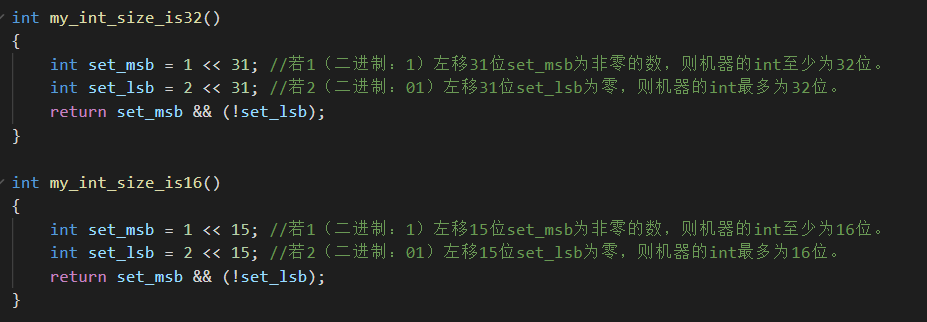


测试样例：





2.67

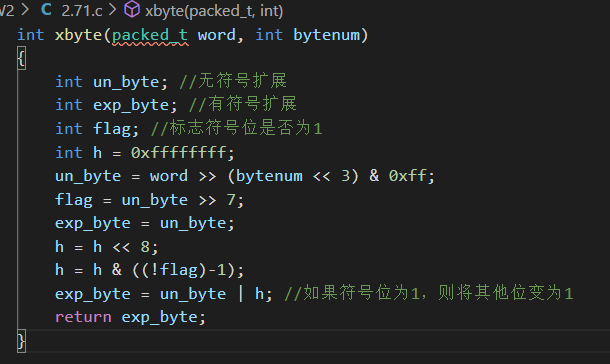


2.71

答：

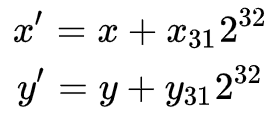
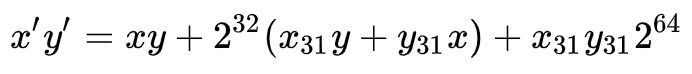
A：原操作将通过将word右移bytenum\*8位将需要的字节移动到最低8位，&0xFF使除了最低的8位其他位都归为0。这种操作不是符号位扩展，而是算数右移，因为这样无论符号位是1还是0，都被扩展为0。

B：



我的方法开始与原方法相同，之后的操作与题目2.63类似，flag为标志位，若需操作字节的符号位为1，则flag为1，否则为0。若flag为1，则h被操作为0xffffff00，无符号扩展后的数与h按位或后即为有符号扩展。若flag为0，则h为0x00000000，无符号扩展与有符号扩展相同，所以不变。

2.75

答：由书中等式2.18 可得，。

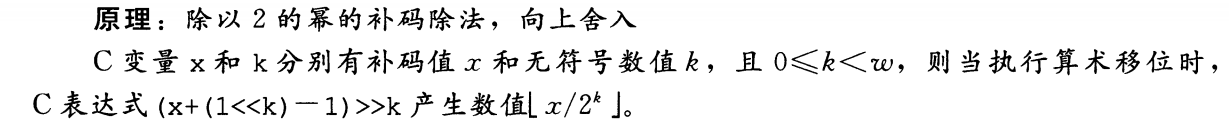
我们要将32位的乘法扩展到64位，则最终的结果只精确263，即最终的结果会进行对264的取模运算。所以我们可以将最后一项舍去，最终的公式变为：

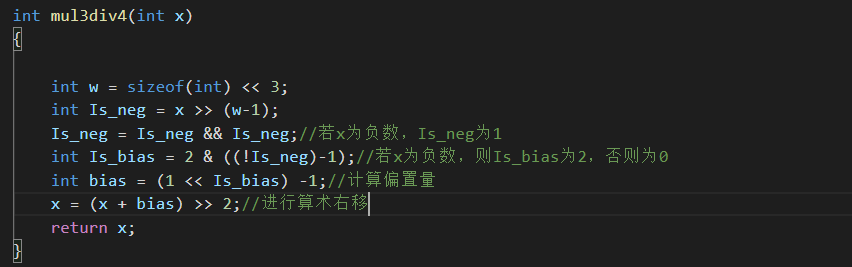


由以上等式编程。



2.79





根据书中补码除以2k得到的原理，设计该算法。

2.83

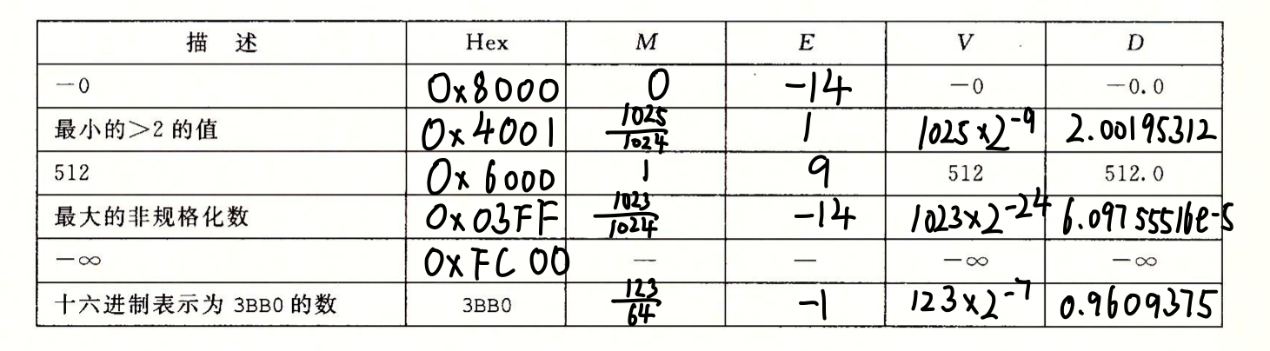
答：A：设X为这个无穷串的值，由X·2k = X+Y可得。X = Y/2k-1。

B：（a）5/7

（b）2/5

（b）19/63

2.87



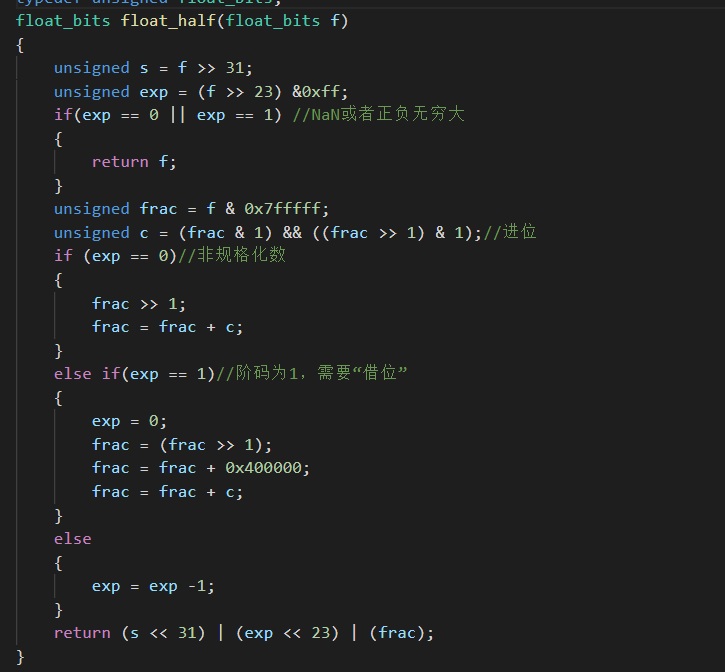
2.91

答：A：11.0010010000111111011011

B：11.001001(001循环)…

C：第九位，0x40490FDB的第九位为0，22/7的二进制小数的第九位为1。

2.95



分类讨论：

1. 当浮点数为NaN或者无穷大时，直接返回其本身。
2. 当浮点数为非规格化数时，要计算是否进位。

是否进位需要观察frac的最后两位XY，若Y是1，向右移位1位后则需要舍入。

向偶数舍入要使最低位为0，若X=1，则frac+1；若X=0，则直接舍去Y。

1. 当浮点数为规格化数时，直接将exp减去1即可。