**2018 春第 2 章家庭作业**

2018.4.12

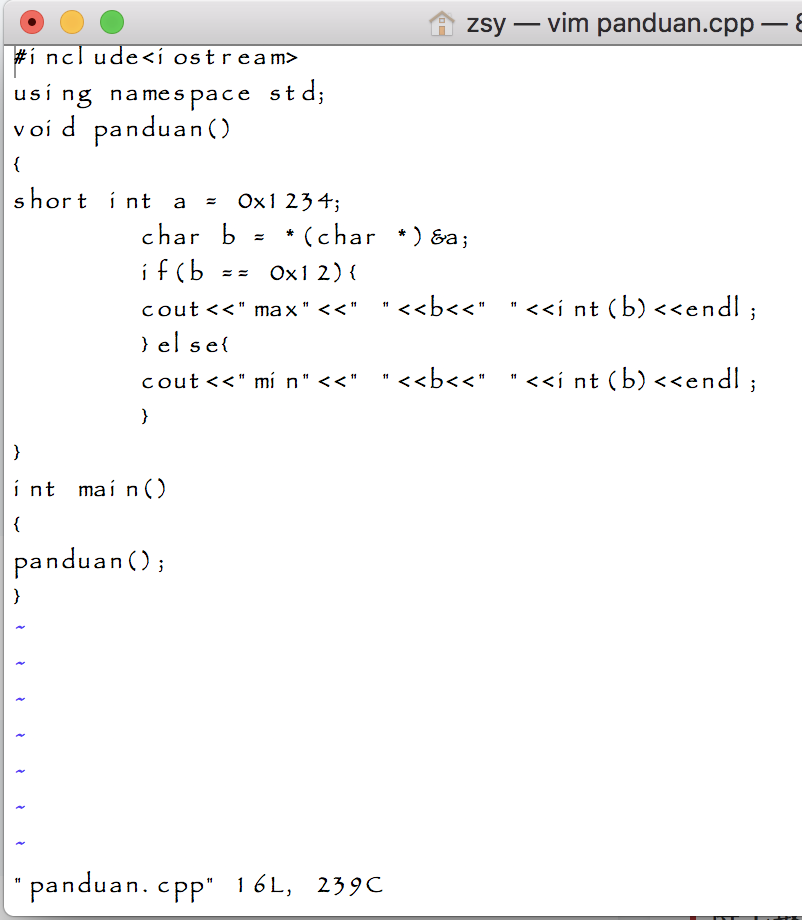
1. **2.58 编写过程 is\_little\_endian，当在小端格式机器上编译和运行时返回 1，在大 端格式机器上编译运行时则返回 0。这个程序应该可以运行在任何机器上，无论机器的字长是多少。**

分析：

**大端**：低位高地址，高位低地址。

**小端**：高高低低。

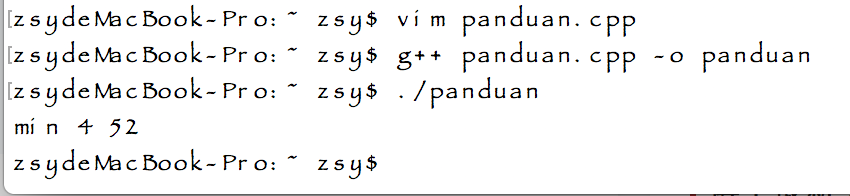
代码截图：



如果是小端存储，那么输出的数字就应该是

b=4（低位存在低地址）

int（b）=52，也就是0x34的十进制



1. **2.71 你刚刚开始在一家公司工作，他们要实现一组过程来操作一个数据结构，要将4个有符号字节封装成一个32位unsigned。一个字中的字节从0(最低有效字节)编号到3(最高有效字节)。分配给你的任务是:为一个使用补码运算和算术右移的机器编写一个具有如下原型的函数:**

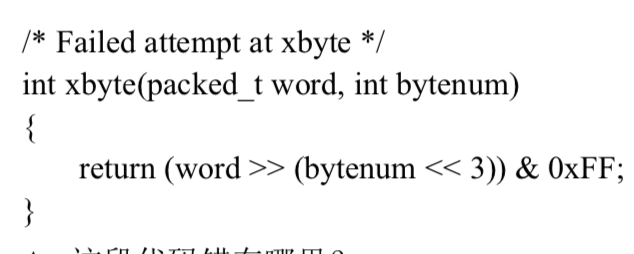
**/\* Declaration of data type where 4 bytes are packed into an unsigned \*/**

**typedef unsigned packed\_t;**

**/\* Extract byte from word. Return as signed integer \*/**

**int xbyte(packed\_t word, int bytenum)**

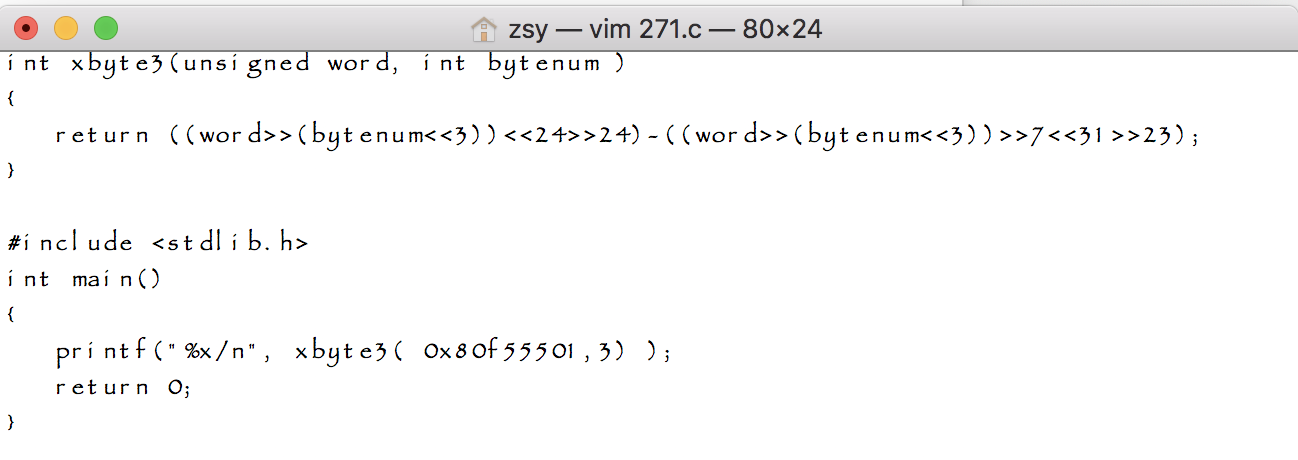
**也就是说，函数会抽取出指定的字节，再把它符号扩展为一个 32位int。你的前任(因为水平不够高而被解雇了)编写了下面的代码:**

****

1. **这段代码错在哪里?**

符号位没有扩展。

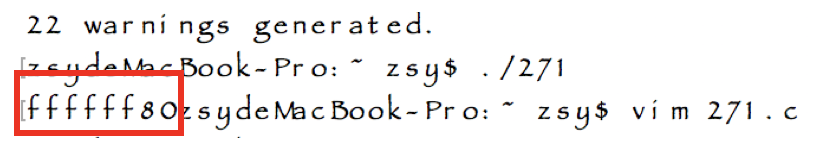
**B. 给出函数的正确实现，只能使用左右移位和一个减法。**



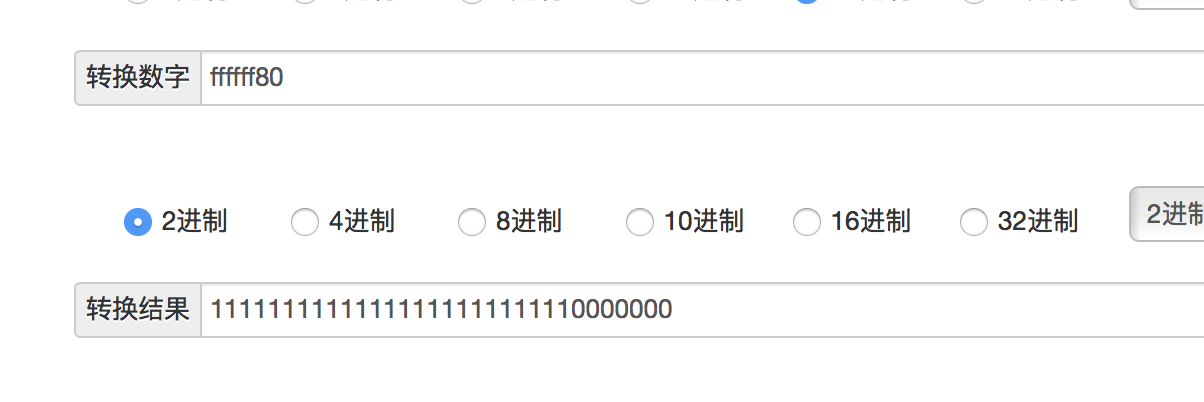
测试样例：



运行结果



检验结果：



答案正确。

第二种方法，把目标字节左移到最高位，用有符号的格式存储，再右移

int xbyte(unsigned word, int bytenum)

{

int a = word << ((3 - bytenum)<<3);

return a >> 24;

}

1. **2.84 给定一个浮点格式，有k位指数和n位小数，对于下列数，写出阶码E、尾数M、小数f和值V的公式。另外，请描述其位表示。**
2. **数 5.0。**

5.0=

E:2

M：左移两位，( )

f:

e=+ 2 =+1。

0（符号位） 10..01（k位） 0100..0（n）

1. **能够被准确描述的最大奇整数。**

E:n

M：1.111111111（n）

f: 111111111111全是1

e=+ n

0（符号位）11111111（k位） 1111111111（n）

V:-1。

1. **最小的正规格化数的倒数。**

最小正规格化数：0（符号位）000000001（k位） 000000000001（n）

倒数为：

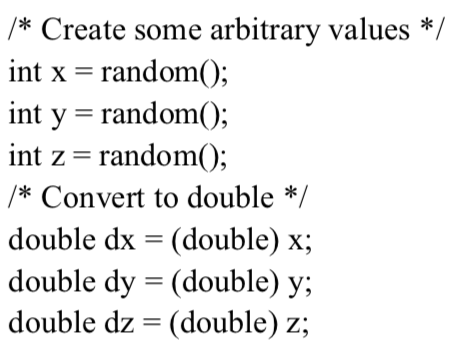
M：1.00000000（n）

f: 00000000000全是0

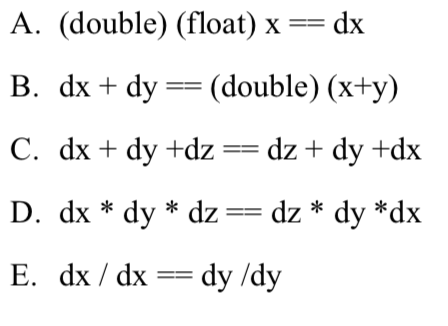
e=+

0（符号位）11111101（k位） 000000000000（n）

1. **2.88 我们在一个 int 类型为 32 位补码表示的机器上运行程序。float 类型的值使用32位IEEE格式，而double类型的值使用64位 IEEE格式。我们产生随机整数x、y和z，并且把它们转换成double 类型的值:**

****

**对于下列的每个 C 表达式，你要指出表达式是否总是为 1。如果它总是为 1，描 述其中的数学原理。否则，列举出使它为 0 的参数的例子。请注意，不能使用IA32机器运行GCC来测试你的答案，因为对于float和double，它使用的都是80位的扩展精度表示。**



A. 0。Float精确度要小于double，所以不会一直相等。

B. 0。比如x,y都是在最大边界的时候，他俩相加很有可能进位，导致进位，而这个新的数会超过double的界限了（左边不会越界）。

C. 1。交换顺序相加不会影响到什么。

D. 0。举个例子，假如Dx\*dy越界的话，先这俩相乘再乘dz不一定会等于dz先乘dy再乘dx。

E. 0。比如，dx为0，dy是5。那么一个结果是nan，另外一个是1。

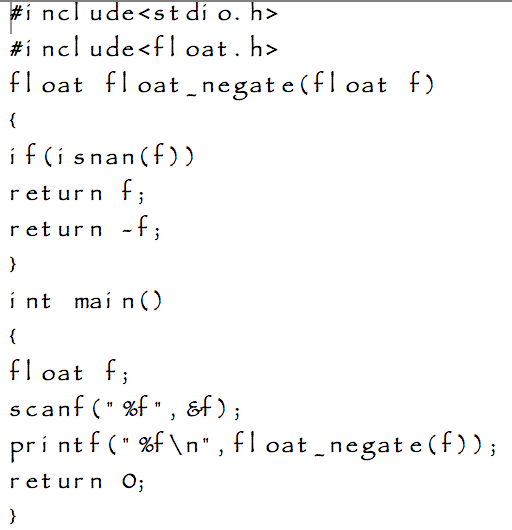
1. **2.92. 遵循位级浮点编码规则，实现具有如下原型的函数: /\* Compute –f. If f is NaN, then return f. \*/**

**float\_bits float\_negate(float\_bits f);**

**对于浮点数 f，这个函数计算- f 。如果 f 是 NaN，你的函数应该简单的返回f。 测试你的函数，对参数f可以取的所有232 个值求值，将结果与你使用机器的浮点运算得到的结果相比较。**

代码：

//isnan函数判断是否为nan



运行截图：

