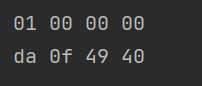
2.56



#include <stdio.h>



typedef unsigned char \*byte\_pointer;

void show\_bytes(byte\_pointer start, size\_t len);

void show\_int(int x);

void show\_float(float x);

int main() {

int i=1;

show\_int(i);

float f = 3.1415926;

show\_float(f);

return 0;

}

void show\_bytes(byte\_pointer start, size\_t len){

size\_t i;

for(i=0;i<len;i++)

printf(" %.2x",start[i]);

printf("\n");

}

void show\_int(int x){

show\_bytes((byte\_pointer)&x,sizeof (x));

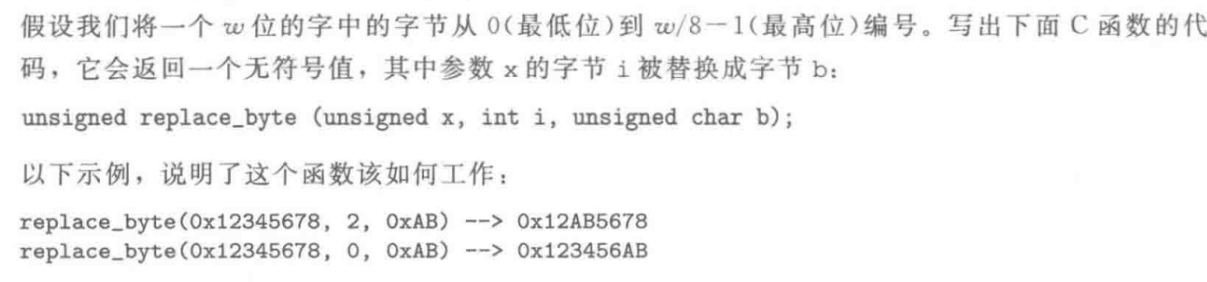
}

void show\_float(float x){

show\_bytes((byte\_pointer)&x,sizeof (x));

}

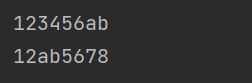
2.60



#include <stdio.h>

unsigned replace\_byte (unsigned x, int i, unsigned char b);

int main() {

 printf("%x\n", replace\_byte(0x12345678,2,0xAB));

printf("%x\n", replace\_byte(0x12345678,0,0xAB));

return 0;

}

//小端机有效

unsigned replace\_byte (unsigned x, int i, unsigned char b){

int a = x;

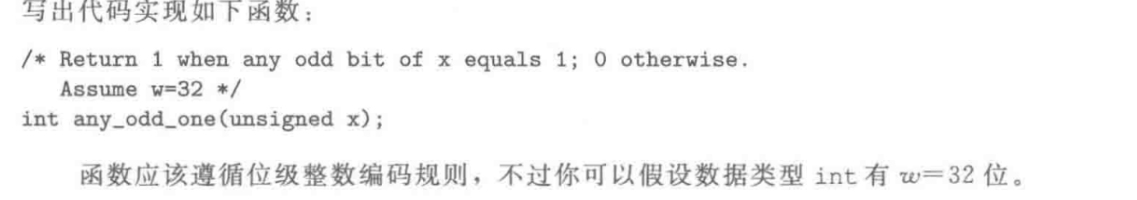
unsigned char \* p = &a;

p[i]=b;

return a;

}

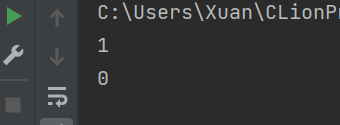
2.64



#include <stdio.h>

int any\_odd\_one(unsigned x);

int main() {

 unsigned a = 0x3;

printf("%d\n",any\_odd\_one(a));

unsigned b = 0x4;

printf("%d\n",any\_odd\_one(b));

return 0;

return 0;

}

/\*Return 1 when any odd bit of x equals 1;0 otherwise

\* Assume w=32

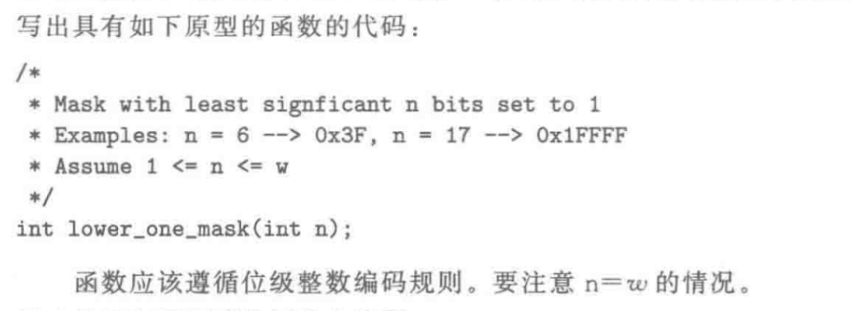
\* \*/

int any\_odd\_one(unsigned x){

return !!(x & 0xaaaaaaaa);

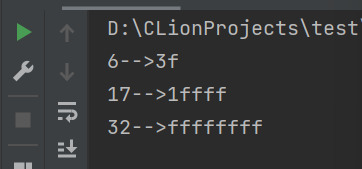
}

2.68



#include <stdio.h>

int lower\_one\_mask(int n);

int main() {

printf("%d-->%x\n",6, lower\_one\_mask(6));

printf("%d-->%x\n",17, lower\_one\_mask(17));

printf("%d-->%x\n",32, lower\_one\_mask(32));

return 0;

}

/\*\*

\* Mask with least significant n bits set to 1

\* Examples: n = 6 --> 0x3F, n = 17 --> 0x1ffff

\* Assume 1<= n <=w

\* \*/

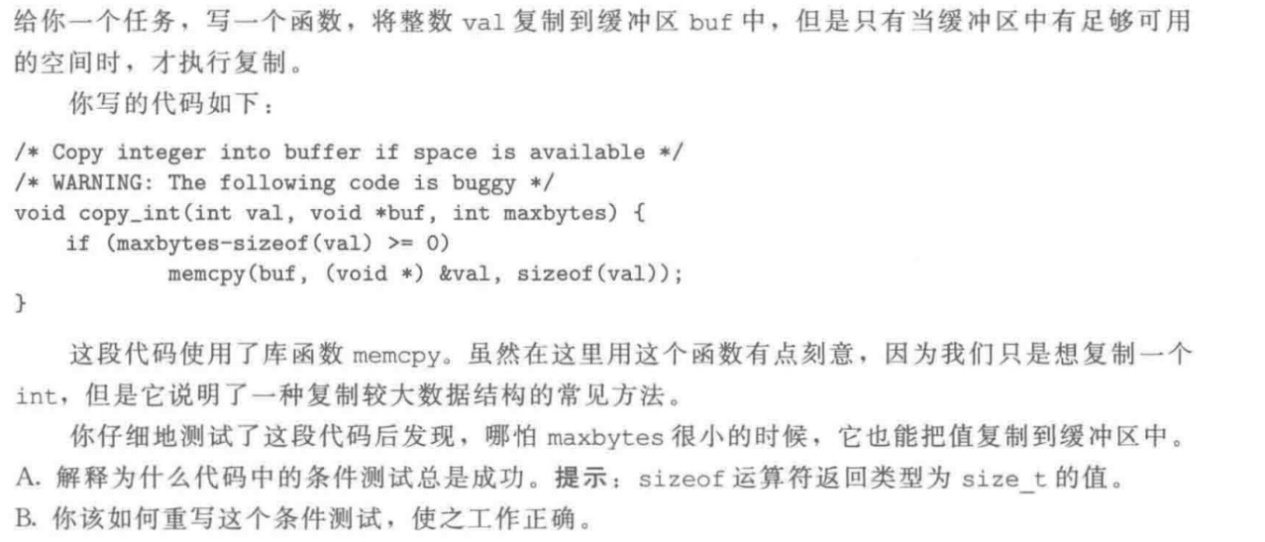
int lower\_one\_mask(int n){

int w = sizeof(int) << 3;

return (unsigned)-1 >> (w - n);

}

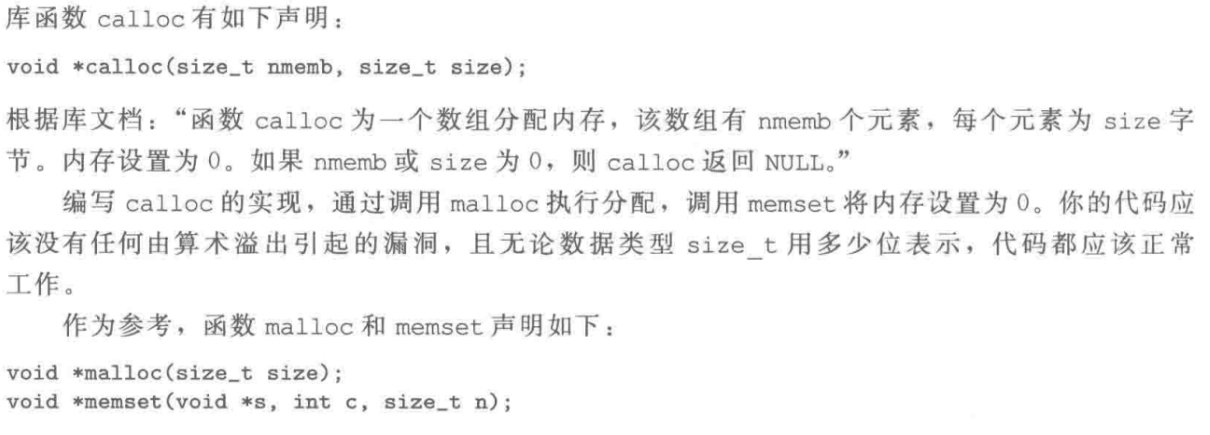
2.72



A．sizeof()的结果是一个unsigned int，当计算maxbytes-sizeof(val) >= 0时，若maxbytes是一个小于等于sizeof(val)的值，那么最后计算的值将是一个很大的unsigned int，将会执行if中的语句，但是在该情况下，本不应该执行。

B．将函数参数中的maxbytes改为unsigned类型

2.76



void \*mycalloc(size\_t nmemb, size\_t size){

if(nmemb==0||size==0)return NULL;

else{

size\_t size1 = nmemb\*size;

if(size1/nmemb==size){

void\* p =malloc(size1);

if(p!=NULL)

memset(p,0,size1);

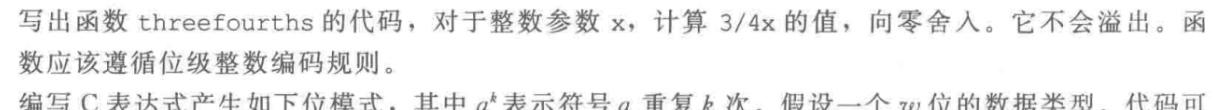
return p;

}

}

}

2.80



int threefourths(int x) {

//判断x正负

int neg\_flag = x & INT\_MIN;

//取x的前30位

int m30 = x & ~0x3;

//取x的后2位

int l2 = x & 0x3;

//计算m30除以4乘以3

int m30d4m3 = ((m30 >> 2) << 1) + (m30 >> 2);

int bias = 3;

//计算l2乘以3

int l2m3 = (l2 << 1) + l2;

//如果x为负

if(neg\_flag)l2m3 = l2m3 + bias;

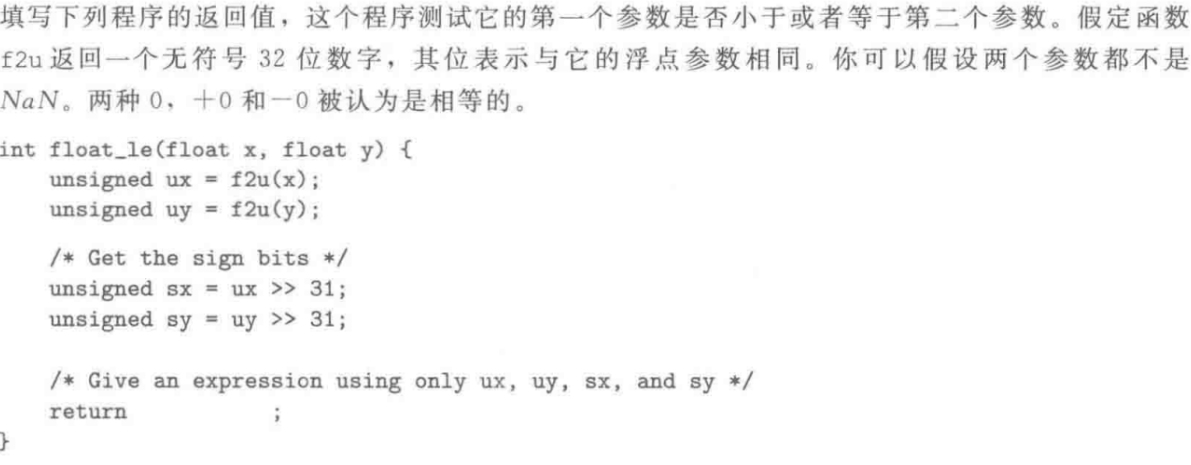
//计算l2乘以3除以4

int l2m3d4 = l2m3 >> 2;

return m30d4m3 + l2m3d4;

}

2.84



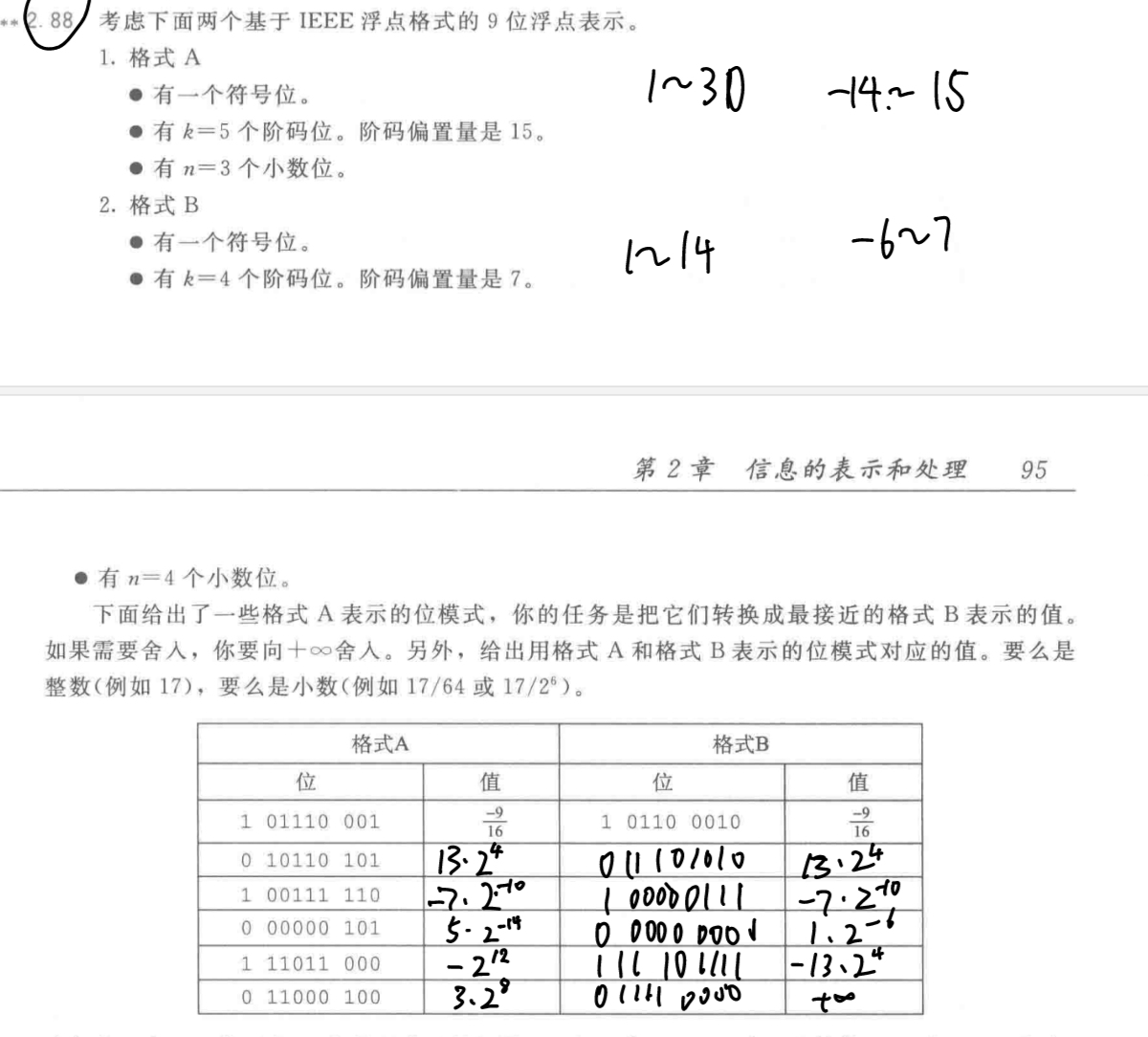
return (ux << 1 == 0 && uy << 1 == 0) || /\* x = y = 0 \*/

(sx && !sy) || /\* x<0 , y>0 \*/

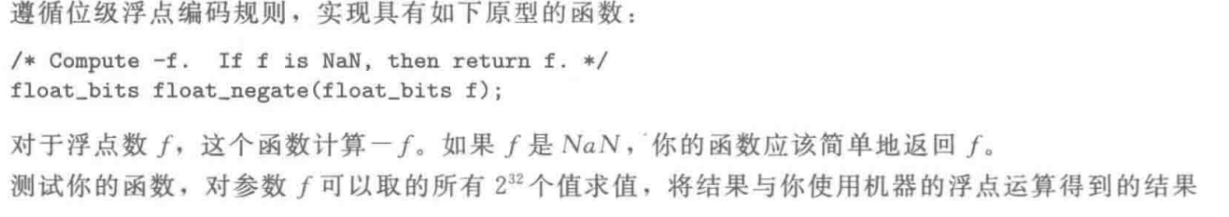
(sx && sy && ux >= uy) ||

(!sx && !sy && ux <= uy);

2.88



2.92





typedef unsigned float\_bits;

float\_bits float\_negate(float\_bits f) {

//符号位

unsigned sign = f >> 31;

//阶码

unsigned exp = f >> 23 & 0xFF;

//尾数

unsigned frac = f & 0x7FFFFF;

//是否NaN

int is\_nan = (exp == 0xFF) && (frac != 0);

if (is\_nan) {

return f;

}

return (~sign << 31) | (exp << 23) | frac;

}