第二章作业

2.61

```c
#include <stdio.h>
#include <stdbool.h>

int A(int x)
{
    return !(~x);
}


int B(int x)
{
    return !x;
}

int C(int x)
{
    return A(x | ~0xff); //和0xff ff ff 00位或看是否全1
}

int D(int x)
{
    return B((x>> ((sizeof(int) - 1) << 3))&0xff); //右移24位观察是否全0
}

int main(){
    //test A
    int a0 = 0xffffffff;
    int a1 = 0xff;
    int a2 = 0xff002;
    printf("x的任何位都等于1\n");
    printf("A(%#x):%d\n",a0, A(a0));   // %# 输出进制前缀
    printf("A(%#x):%d\n", a1, A(a1));
    printf("A(%#x):%d\n\n", a2, A(a2));
    //test B
    int b0 = 0x0;
    int b1 = 0x1;
    int b2 = 0xff;
    printf("x的任何位都等于0\n");
    printf("B(%#x):%d\n", b0, B(b0));
    printf("B(%#x):%d\n", b1, B(b1));
    printf("B(%#x):%d\n\n", b2, B(b2));
    //test C
    int c0 = 0x00ff;
    int c1 = 0x01234;
    int c2 = 0x012345ff;
    printf("x的最低有效字节中的位都等于1\n");
    printf("C(%#x):%d\n", c0, C(c0));
    printf("C(%#x):%d\n", c1, C(c1));
```

```
    printf("C(%#x):%d\n\n", c2, C(c2));
    //test D
    int d0 = 0x00123456;
    int d1 = 0x12345678;
    int d2 = 0xff000000;
    printf("x的最高有效字节中的位都等于0\n");
    printf("D(%#x):%d\n", d0, D(d0));
    printf("D(%#x):%d\n", d1, D(d1));
    printf("D(%#x):%d\n", d2, D(d2));
    return 0;
}
```

2.71

```
A. word是unsigned类型所以右移时会进行无符号扩展，比如原字节是0xff(-1)，这个函数将返回0xff(255)，而非题目要求
B. 先右移 再左移即可保留符号位
int xbyte(packed_t word,int bytenum)
{
    return ((int)(word<<((3-bytenum)<<3)))>>24;
}
```

2.87

格式A		格式B	
位	值	位	值
1 01110 001	$\frac{-9}{16}$	1 0110 0010	$\frac{-9}{16}$
0 10110 101	208	0 1110 1010	208
1 00111 110	-7/1024	1 0000 0111	-7/1024
0 00000 101	5/2^17	0 0000 0001	2^(-10)
1 11011 000	-2^12	1 1110 1111	-248
0 11000 100	768	0 1111 0000	无穷大

2.88

```
A.不总为1
当x超出float能够精确表达的范围时，int转为float会有精度丢失，此时不成立
例如，当x=2^31-1时,此时x转为float即为无穷大，不成立

B.不总为1
x=Tmax,y=Tmax,x+y溢出得到负数，而dx+dy由于是double类型，不溢出，导致二者不相等

C.总为1
double可精确表示2^53内的所有整数，而3个最大值为（2^31-1）的数相加不可能超出这个范围

D.不总为1
3个最大值为（2^31-1）的数相乘可能导致结果超出2^53，会导致舍入，造成精度丢失，不成立
```

E.不总为1
当dx或dy为0时，显然不成立