1. ex 2.10

步骤	*x	*y
初始	a	b
第一步	a	a^b
第二步	a^(a^b)=0^b=b	a^b
第三步	b	b^(a^b)=0^a=a

2. ex 2.11

A. first= last= k;

B. 在最后一次 for 循环时，left == right == k . inplace_swap 函数参数传递为 inplace_swap(&a[k],&a[k]), 此时在这个函数中，x 和 y 地址是相同的。改变 *y 也会改变 *x ，在执行按位异或时，会把 *x 和 *y 都置为 0.

C. 修改函数 inpalce_swap(int *x, int *y) ,在函数汇总判断 x 和 y 是否指向相同地址，若不指向同一地址，函数继续，否则函数返回。

3. ex 2.14

	机器数 (十六进制)		机器数 (十六进制)
x&y	0x20	x&&y	0x01
x|y	0x7f	x||y	0x01
~x | ~y	0xdf	!x || !y	0x00
x & !y	0x00	x && ~y	0x01

4. ex 2.23

w	fun1(w)	fun2(w)
0x00000076	0x00000076	0x00000076
0x87654321	0x00000021	0x00000021
0x000000C9	0x000000C9	0xFFFFFFC9
0xEDCBA987	0x00000087	0xFFFFFF87

5. ex 2.35

当 x=0，容易验证程序的正确性。当 x≠0 时：

1) 可知 w 位的补码数 x，y 相乘最多产生一个有 2w 位的乘积。将这个乘积写成

$$x * y = p + t2^w$$

其中 p 表示低 w 位表示的数值，$t2^w$表示高 w 位的数值（将低 w 位置为 0）。易知当且仅当 t≠0 时 p 的计算溢出。

2) 设 P 是一个有 2w 位的数，且 P=x*y。设 Q 是一个有 w 位的数，且 Q=P/x，有：

$$P = x * Q + r$$

当 p 没有溢出时，也可以写成这个表达式。

3) 由 1) 和 2) 中的两个式子联立可得

$$x * (y - q) = r + t2^w$$

当 y≠q 时，有$r + t2^w \neq 0$，此时 r 和 t 不全为 0. 故当$r + t2^w = 0$时，即 r = t = 0 时，y=q。

6. ex 2.47

位	e	E	2^E	f	M	2^E*M	V	十进制
00000	0	0	1	0/4	4/4	4/4	4/4	1.00
00001	0	0	1	1/4	5/4	5/4	5/4	1.25
00010	0	0	1	2/4	6/4	6/4	6/4	1.50
00011	0	0	1	3/4	7/4	7/4	7/4	1.75
00100	1	0	1	0/4	4/4	4/4	4/4	1.00
00101	**1**	**0**	**1**	**1/4**	**5/4**	**5/4**	**5/4**	**1.25**
00110	1	0	1	2/4	6/4	6/4	6/4	1.50
00111	1	0	1	3/4	7/4	7/4	7/4	1.75
01000	2	1	2	0/4	4/4	8/4	8/4	2.00
01001	2	1	2	1/4	5/4	10/4	10/4	2.50
01010	2	1	2	2/4	6/4	12/4	12/4	3.00
01011	2	1	2	3/4	7/4	14/4	14/4	3.50
01100	--	--	--	--	--	--	∞	--
01101	--	--	--	--	--	--	NaN	--
01110	--	--	--	--	--	--	NaN	--
01111	--	--	--	--	--	--	NaN	--

7. ex .2.52

格式 A		格式 B	
位	值	位	值
011 0000	1	0111 000	1
101 1110	15/2	1001 111	15/2
010 1001	25/32	0110 100	3/4
110 1111	31/2	1100 000	32
000 0001	1/64	1101 000	1/64

8. ex 2.63

```
1.  unsigned srl(unsigned x, int k)
2.  {       /*Perform shift arithmetically*/
3.      unsigned xsra = (int)x >> k;
4.      int w = sizeof(x) << 3;
5.      int m = (1 << (w - k)) - 1;
6.      return m & xsra;
```

```
}

int sra(int x, int k)
{    /*Perform shift logically*/
     int xsrl = (unsigned)x >> k;
     int w = sizeof(x) << 3;
     int m = 1 << w - k - 1;
     if (m & xsrl) {
          m << 1, m -= 1;
          xsrl = xsrl | ~m;
     }
     return xsrl;
}
```

9. ex 2.75

```
unsigned unsigned_high_pprod(unsigned x, unsigned y)
{
     return signed_high_prod(x, y) + (x >> (w - 1))*y + x*(y >> (w - 1));
}
```

10. ex 2.81

A. 否。若 y=-（1<<31)。对任意大于 y 的 x 值，-x<-y 并不满足。
B. 是。
C. 否。x=7，y=5.
D. 是。Int 类型转 unsigned 类型编码没有变化。
E. 是。如果有舍入只会是左边的值变小；

11. ex 2.86

描述	Hex	M	E	V
-0	0x8000	0	-62	--
最小的值>1	0xfeff	32	6	-505/2
256	0x4700	1	8	--
最大的非规格化数	0x00ff	255/256	-62	$255*2^{-70}$
-∞	0xff00	--	--	--
十六进制表示为 3AA0 的数	--	13/8	-5	13/256

12. ex 2.90

A. 13176759/8388608
B. 11.001001001001001…
C. 第九位