Homework 3

1. In C language, if an evaluation expression contains both unsigned and signed values, then signed values will be implicitly casted into unsigned ones before evaluation. Please fill the following table with "<", ">" or "=". (Assume **int value is encoded using 16 bits**)

Constant A	Constant B	A ? B
-2U	-1U	<
-1	1	<
-1	100U	>
-1	65535U	=
-32767	32768U	>

2. There is a illustration of code vulnerability similar to that found in FreeBSD's implementation of getpeername(). Find one bug in the following codes and try to fix it.

```
/* Copy n bytes from src to dest */
/* Note: size_t means unsigned int */
void *memcpy(void *dest, void *src, size_t n);

/* Kernel memory region holding user-accessible data */
#define KSIZE 1024
char kbuf[KSIZE];

/* Copy at most maxlen bytes from kernel region to user buffer */
/* Must not copy more than maxlen bytes */
  int copy_from_kernel(void *user_dest, int maxlen) {
     /* Byte count len is minimum of buffer size and maxlen */
     int len = KSIZE < maxlen ? KSIZE : maxlen;
     memcpy(user_dest, kbuf, len);
     return len;
}
```

When we call `copy_from_kernel`, if we set `maxlen` to -1 (or other minus number), `len` will equal `to -1`. Then when we call `memcpy`, it will convert `len` (-1) to `size_t (unsigned int)` implicitly, which would be a huge number. So it could copy a large area from kernel to user, which is very dangerous.
You can check `maxlen` more carefully, like forbidding `maxlen < 0`, set `KSIZE` as `1024u`, unify type of length...

3. Assume x and y are both 4 bit signed integers. Fill the following table. Truncate all the results to 4 bits with 2's complement and write their value in decimal.

	x+y	x-y	x*y	-y
x=4, y=7	-5	-3	-4	-7
x=-6, y=-8	2	2	0	-8
x=5, y=-1	4	6	-5	1
x=-3, y=6	3	7	-2	-6