When using the `%` operator with operands with different signs, the sign of the result is implementation defined. From the documentation: `If both operands are nonnegative then the remainder is nonnegative; if not, the sign of the remainder is implementation-defined.` For example, this is the result of a remainder operation in C++: `(-7 % 3) = -1` In general, when dividing a negative number by a positive number, the quotient will be negative (or zero). This happens due to the way x86 architectures perform division, which generates a negative remainder. If you want to enforce positive result, as in the mathematical definition of modulus, you should do something like the following:

```
inline int mod(int a, int b) {
    int ret = a % b;
    return ret >= 0 ? ret : ret + b;
}
```

Or using templates:

```
template<typename T>
inline T mod(T a, T b) {
    T ret = a % b;
    return ret >= 0 ? ret : ret + b;
}
```