3) (5 pts) ALG (Bitwise Operators)

There are a total of 25 cards, numbered 0 through 24. We can represent a set of cards with a single integer by setting the i^{th} bit to 1 if the set contains card i, and setting the bit to 0 otherwise. For example, the set of cards {2, 6, 7} would be stored as the integer 196, since $196 = 2^7 + 2^6 + 2^2$. Two sets of cards are disjoint, if and only if no card appears in both sets. Complete the function below so that it returns 1 if the sets of cards represented by the integers set1 and set2 are disjoint, and returns 0 if they are not disjoint. (For example, disjoint(196, 49) should return 1 because $49 = 2^5 + 2^4 + 2^0$, and there is no common value in the two sets {2, 6, 7} and {0, 4, 5}. On the other hand, disjoint(196, 30) should return 0 because $30 = 2^4 + 2^3 + 2^2 + 2^1$, so that card number 2 is included in both sets 196 and set 30.)

```
// Pre-condition: set1 and set2 are bitmasks in between 0 and
//                (1<<25)-1.
// Post-condition: Returns 1 if the two bitmasks are disjoint,
//                 meaning that the sets they represent don't have
//                 any items in common, and returns 0 otherwise, if
//                 the two represented sets do have common items.
int disjoint(int set1, int set2) {

    return (set1 & set2) == 0;

}
```

Grading: 1 pt return
3 pts and between sets
1 pt checking if that and is 0 or not.

Note: Many other ways to do this that are less succinct. As long as bitwise ops are used, give full credit even if there is a loop (25 times) that looks only at pairs of bits at a time and does the right thing. So something like this would get full credit:

```
int disjoint(int set1, int set2) {

    for (int i=0; i<25; i++) {
        if ( (set1&1) && (set2&1) ) return 0;
        set1 >>= 1;
        set2 >>= 1;
    }

    return 1;

}
```