3) (10 pts) ALG (Backtracking)

Imagine a dating website that asks each user 20 yes/no questions and stores their answers in a single integer in between 0 and 2^{20}-1, with the answer to the kth question stored in bit 2^{k-1}, with 0 representing no and 1 representing yes. (So for example, if one person's answers to questions 1, 3 and 4 were yes and the rest of the answers were no, the integer $1101_2 = 13$ would be used to represent her 20 answers to the questions.) Consider the problem of finding the "best match" for a client out of a list of prospective matches. We consider a match **A** for the client to be better than match **B** if she shares more answers on corresponding questions with **A** than **B**. Write a function that takes in an integer representing the client's answers, an array of integers storing the answers of potential matches, and the length of that array, which returns the index into the array storing the best match for that client. If there are multiple best matches, you may return the index associated with any of them. A function has been given that you may call in your code, if you find it useful.

```
int bestMatch(int client, int* matches, int length) {

    int res = 0, best = -1, i;

    for (i=0; i<length; i++) {
        int tmp = 20 - count(client ^ matches[i]);
        if (tmp > best) {
            best = tmp;
            res = i;
        }
    }

    return res;

}

int count(int mask) {
    int res = 0, i;
    for (i=0; i<32; i++)
        if ((mask & (1<<i)) != 0)
            res++;
    return res;
}
```

Grading:
1 pt for var declarations and initializations,
1 pt for for loop to length (can start at 0 or 1 depending on initializations),
3 pts for the appropriate XOR (note that we don't have to do 20 -…we can simply choose to minimize the number of bits on in the xor),
2 pts for the function call to count,
2 pts for the mechanism to select the appropriate index depending on the result of the function call,
1 pt for the return