1.) We wish to develop an algorithm to determine if a list of integers has any duplicated elements. Our first attempt is the following:

```c
IsUnique(array A, length N) {
    int j, k;
    for (j = 1; j <= N; j++) {
        for (k = 1; k <= N; k++) {
            if (j != k && A[j] == A[k])
                return false;
        }
    }
    return true;
}
```

IsUnique checks each pair of values twice, which is unnecessary. To deal with this, we develop the following algorithm:

```c
IsUnique2(array A, length N) {
    int j, k;
    for (j = 1; j <= N-1; j++) {
        for (k = j+1; k <= N; k++) {
            if (A[j] == A[k])
                return false;
        }
    }
    return true;
}
```

a.) What is the tightest Big-Oh of “IsUnique”? (Remember: consider the worst-case)

b.) What is the tightest Big-Oh of “IsUnique2”? 

c.) Assume that you have a sort routine that will take nlog₂n time to sort the values in the array. Develop a more efficient IsUnique algorithm that will use this routine, and give the tightest Big-Oh for this new routine – note: the call to the sort routine would be part of your program, and thus its execution time would be part of your runtime.

2.) For the following problems, state whether we know the problems are NP-complete, NP-hard, P, NP, or unknown, given the information specified. Be as specific as possible (i.e. if something is NP-Complete and you just say it is in NP, this is not as specific as you can be). Careful – these are trick questions.

Note: “map efficiently” and “transform” are synonyms for “can be converted in polynomial time”.

a.) CHOP is NP-complete. We can map CHOP efficiently to SPINDLE, and SPINDLE efficiently to MUTILATE. MUTILATE is in NP. What do we know about SPINDLE and MUTILATE?
b.) We can efficiently map TWEEDLE-DEE to TWEEDLE-DUMB, and TWEEDLE-DUMB is in NP. What do we know about TWEEDLE-DEE and TWEEDLE-DUMB?

c.) We can efficiently transform YIN into the NP-hard problem YANG. What do we know about YIN?

d.) JOHN and PAUL are NP-complete problems, and RINGO is in P. We can efficiently transform JOHN into RINGO. What do we know about JOHN, PAUL, and RINGO?

e.) JABBERWOCKIE is NP-complete. We can efficiently map JABBERWOCKIE into JUBJUB. What do we know about JUBJUB?

3.) 4-SAT and 5-SAT are similar to 3-SAT, but with 4 and 5 terms respectively in each Sum Term. 5-SAT is NP-complete. What is the complexity of 4-SAT? You must give a complete proof of your answer.