1 Algorithm Analysis

A sorting algorithm is stable provided that for any pair of elements \( A[i] \), \( A[j] \) in the input array \( A \) with \( A[i] = A[j] \) and \( i < j \), these elements are in the sorted array at indices \( i' \), \( j' \) with \( i' < j' \).

**Problem 1.** Is the following (insertion-sort-based) sorting algorithm stable?

```c
void sort(int* A, int n)
{
    for (int i = 0; i < n; ++i)
        for (int j = i-1; j >= 0; --j)
            if (A[j] >= A[j+1])
                {
                    int tmp = A[j];
                    A[j] = A[j+1];
                    A[j+1] = tmp;
                }
}
```

If it is not stable, how could it be modified to be stable?

**Problem 2.** Analyze the running time (on inputs of length \( n \)) of a radix sorting algorithm where the input is given in binary (i.e., each “digit” is a a bit with value 0 or 1) and comes from the universe \( \{0, 1, \ldots, 2^p - 1\} \).

2 Algorithm Implementation

**Problem 3.** Implement a C++ function that uses counting sort to sort an array \( A \) (of length \( n \)) consisting of Entry objects according to their rating values (guaranteed to be between 0 and 9, inclusive):

```c
class Entry
{
    public:
        string hashtag;
        int rating;
};
```
3 Algorithm Design

A sequence is *binary* if it consists of only 0s and 1s.

**Problem 4.** Give an $O(n)$-time algorithm that sorts a binary array $A$ (of length $n$) and only modifies elements via the `swap` function.