**Problem 1.** Let $A$ be an array of 6 integers as follows: $(8, 3, 4, 1, 7, 10)$. Suppose that we use counting sort to sort the array, knowing that all the integers are in the domain from 1 to 10. Recall that the algorithm (as described in the class) generates an array $B$ where each element is either 0 or 1. Enumerate the elements of $B$ by indicating which ones are 0.

**Problem 2.** Describe the output of each `pop()` operation in the following sequence of operations on an initially empty stack:

```
push(56), push(6), push(83), pop(), push(15), pop(), pop().
```

**Problem 3.** Describe the output of each `de-queue()` operation in the following sequence of operations on an initially empty stack:

```
en-queue(56), en-queue(6), en-queue(83), de-queue(), en-queue(15), de-queue(), de-queue().
```

**Problem 4.** Let $A$ be an array of $n$ integers already sorted in ascending order. Let $B$ be array of $m$ integers that are not sorted. We know that the set of integers in $A$ is disjoint with the set of integers in $B$. Describe an algorithm to produce an array where all the $n + m$ integers have been sorted in ascending order. Your algorithm should terminate in $O(n + m \log m)$ time.

**Problem 5.** Consider a sequence of $n$ brackets, where each bracket is either opening (namely “[”) or closing (“]”). The sequence is legal if, intuitively, every opening bracket finds its closing counterpart. For example, `[[]]` is legal but `[[]` is not. Formally, a legal sequence is such that, one can continuously remove two adjacent brackets `[]` until all the brackets have disappeared. Suppose that the sequence is stored in an array of length $n$, where each bracket is stored in a cell. Give an algorithm to check whether the sequence is legal in $O(n)$ time.

**Problem 6.** Suppose that we want to support two operations on an initially empty multi-set $S$:

- **Insert($e$):** which inserts an integer $e$ into $S$
- **Query($e$):** which asks whether integer $e$ belongs to $S$.

We will need to process a sequence of operations that consists of $n$ insertions, then followed by $n$ queries. These operations are given to us one by one, namely, the next operation is given only after the previous one has been processed. We do not know the value of $n$, until receiving the first query. Design an algorithm that can (i) answer all queries correctly, and (ii) process all the $2n$ operations in $O(n \log n)$ total time.