Your solutions must be submitted to Blackboard as a PDF document.

1. (10 pts.) Using the book’s pseudocode conventions, write a version of the Merge algorithm that does not use sentinels. That is, line 3 becomes:

   Let \( L[1..n_1] \) and \( R[1..n_2] \) be new arrays.

and lines 8 and 9 are deleted. What changes need to be made to lines 12–17? [Hint: You only need to change the condition.]

2. (10 pts.) Using Figure 7.1 as a model, illustrate the operation of the book’s Partition algorithm on the array:

   \[
   1 \ 3 \ 5 \ 7 \ 9 \ 2 \ 3 \ 5 \ 7 \ 5
   \]

3. (10 pts.) In the book’s Partition algorithm, what happens when the pivot value is equal to many elements of the \( A[p..r] \) subarray? What would be the running time of Quicksort with this Partition algorithm if every element in the array was equal to 13? Using the book’s pseudocode conventions, write a revised version of Partition with a small number of changes to handle the issue.

4. (30 pts.) Consider this Randomized-Partition algorithm:

   \[
   \text{RANDOMIZED-PARTITION}(A, p, r)
   \]
   \[
   \begin{aligned}
   &\text{do} \\
   &\quad i = \text{RANDOM}(p, r) \\
   &\quad \text{exchange } A[r] \leftrightarrow A[i] \\
   &\quad q = \text{PARTITION}(A, p, r) \\
   &\quad \text{until } q - p \leq 3 \times (r - p + 1)/4 \text{ and } r - q \leq 3 \times (r - p + 1)/4 \\
   &\text{return } q
   \end{aligned}
   \]

   That is, it requires a partition where neither side is more than \( 3/4 \) of the subarray. [Note: We don’t actually want to use this algorithm, but it is intended to lead to a simplified analysis of quicksort.]

   (a) What is the probability that a call to PARTITION will satisfy the until condition?
   (b) What is the expected number of iterations of the do-until loop?
   (c) Illustrate a recursion tree for a Quicksort algorithm that uses this partition procedure. The recursion tree should be used to derive upper bounds on the expected height of the recursion tree and the expected running time of Quicksort.
5. (10 pts.) Using Figure 6.3 as a model, illustrate the operation of the book’s Build-Max-Heap algorithm on the array:

```
1 3 5 7 9 2 3 5 7 5
```

6. (30 pts.) Consider the following procedure Max-Heap-Check.

```
Max-Heap-Check(A, i)
    if A[i] < A[Parent(i)]
        return false
    if i < A.heap-size
        return true
    else
        return Max-Heap-Check(A, Left(i)) or Max-Heap-Check(A, Right(i))
```

It is supposed to return true if A is a max-heap. The initial call is Max-Heap-Check(A, 1).

(a) The pseudocode contains several bugs. Rewrite the pseudocode so it is correct.

(b) What is the running time of Max-Heap-Check? You need both $O$ and $\Omega$ in this case.

(c) Write and solve a recurrence that models the worst-case running time of Max-Heap-Check.