Homework 1, Solution set

1.a. Step 2 performs 2 operations per iteration (an increment of j and a test of whether $j > n$). Step 3 performs 1 operation per iteration. Step 4 performs 3 operations per iteration (computations of each subcondition, plus the calculation of the and); arguably, this step performs 4 operations, the 4th step being the decision of whether to exit the loop or not. Step 5 performs 3 operations per iteration. Step 6 performs 1 operation per iteration. Step 7 performs 1 operation per iteration as does Step 8.

Step 2 is repeated n times (remember, the exit iteration will see j increased to $n+1$); however, the body of the for loop is executed $n-1$ times. On the jth iteration of the for loop, the while loop is iterated up to $j-1$ times (and so Step 4 is iterated up to j times).

This yields the following upper bound on the number of operations performed.

$$\begin{aligned}
&2n + (n-1) + 3\sum_{j=2}^{n} j + 3\sum_{j=2}^{n}(j-1) + \sum_{j=2}^{n}(j-1) + \sum_{j=2}^{n}(j-1) + n - 1 \\
&\quad = 4n - 2 + \frac{3}{2}(n-1)(n+2) + \frac{5}{2}n(n-1) \\
&\quad = 4n^2 + 3n - 3.
\end{aligned}$$

b. Let the input comprise the values $1, 2, \ldots, n$ in sorted order. Then Step 3 is performed once for each value of j and Steps 5–7 are not executed. This reduces the number of operations to

$$2n + (n-1) + 3\sum_{j=2}^{n} 1 + n - 1 = 4n - 2 + 3(n-1) = 7n - 5.$$

2.a. (i) $f = \Theta(g)$.
b. (ii) $f = O(g)$, not $\Theta(g)$.
c. (iii) $g = O(f)$, not $\Theta(f)$.
d. (ii) $f = O(g)$, not $\Theta(g)$.
e. (ii) $f = O(g)$, not $\Theta(g)$.
f. (i) $f = \Theta(g)$.
g. (iii) $g = O(f)$, not $\Theta(f)$.
h. (ii) $f = O(g)$, not $\Theta(g)$.
i. (iv) None of these.
j. (ii) $f = O(g)$, not $\Theta(g)$.

3.a. We want to show that for every constant $c > 0$ there is a constant n_c such that for all $n \geq n_c$, $f(n) \leq c \cdot g(n)$. Equivalently, we want that $5n^2 + 3n \leq c \cdot 3n^3$. For $n \geq 1$, $5n^2 + 3n \leq 8n^2$, so it suffices that $8n^2 \leq 3cn^3$, or equivalently that $8 \leq 3cn$. This holds whenever $n \geq 3/c$. Thus, for all $n \geq 3/c$, $f(n) \leq c \cdot g(n)$.

b. We want to show that for every constant $c > 0$ there is a constant n_c such that for all $n \geq n_c$, $2^n \leq c \cdot 3^n$. Equivalently, we want that $\frac{1}{c} \leq \left(\frac{3}{2}\right)^n$. Taking logs yields that

$\log \frac{1}{c} \leq n \log \frac{3}{2}$. Let $n_c = \lceil \max\{1, \log \frac{1}{c}\} / \frac{3}{2} \rceil$. (The reason for taking the max is to ensure the first term is positive.) Then, for all $n \geq n_c$, $2^n \leq c \cdot 3^n$.

4.a. We first merge L_1 and L_2 and then merge the result with L_3. The total number of steps performed is $(\ell_1 + \ell_2 + 1) + (\ell_1 + \ell_2 + \ell_3 + 1) = 2\ell_1 + 2\ell_2 + \ell_3 + 2$. If instead, we started by merging L_1 and L_3, the cost would become $2\ell_1 + 2\ell_3 + \ell_2 + 2 \geq 2\ell_1 + 2\ell_2 + \ell_3 + 2$, and if we started by merging L_2 and L_3, the cost would become $2\ell_2 + 2\ell_3 + \ell_1 + 2 \geq 2\ell_1 + 2\ell_2 + \ell_3 + 2$. Thus the given order is the least expensive one.

b. We first merge L_1 and L_2. Let the resulting list be named M. If $|M| < |L_4|$ we next merge M and L_3, and merge the result with L_4. Otherwise, we next merge L_3 and L_4, and merge the result with M. Note that the second stage involves 3 lists, and we are following the rule from part (a), namely to start by merging the shortest two lists. Thus, the only step we need to justify is the initial merge of L_1 and L_2.

The number of steps performed by our procedure is the smaller of $3\ell_1 + 3\ell_2 + 2\ell_3 + \ell_4 + 3$ and $2\ell_1 + 2\ell_2 + 2\ell_3 + 2\ell_4 + 3$; any other ordering of the merges will result in a corresponding permutation of the terms $\ell_1, \ell_2, \ell_3, \ell_4$, which can only increase the total number of steps. Thus the given procedure provides the best order.