1. **Counting operations**

   Count the number of primitive operations on each line of the following method body:

```
public static void minToFront(List<Integer> t) {
    if (t.size() < 2) {
      return;
    }
    Week10.minToFront(t.subList(1, t.size()));
    int first = t.get(0);
    int second = t.get(1);
    if (second < first) {
      t.set(0, second);
      t.set(1, first);
    }
  }
```

2. **Recurrence relations**

   Derive the recurrence relation for the following method:

```
public static void selectionSort (List<Integer> t) {
    if (t.size() > 1) {
        Sort.minToFront(t);
        Sort.selectionSort(t.subList(1, t.size()));
    }
}
```

   Start by counting the number of elementary operations there are on each line, then determine how often each line runs. From there, state the number of elementary operations required by the base case, $T(1)$, and then state the number of elementary operations required by the recursive case, $T(n)$.

3. **Solving recurrence relations**

   Solve the following recurrence relations:

   (a) $T(1) = 1$
   $T(n) = T(n-1) + 3n$

   (b) $T(1) = 7$
   $T(n) = T(n/2) + 1$

4. **Big-O** Prove each of the following:

   (a) $f(n) = n^4 + 3n^2 + n \in O(n^4)$

   (b) $f(n) = 12n + 5n \log_2 n \in O(n \log_2 n)$

   (c) $f(n) = 2^n + 100n^3 \in O(2^n)$, note that $2^n > n^3$ for all $n > 9$

5. **Proof of correctness and termination**

   Prove that the following method is correct and terminates:

```
/**
 * Returns the value of the sum:
 *
 * m + (m + 1) + (m + 2) + ... + n
 *
 * @param m an integer
 * @param n an integer
 * @pre. m < n
 * @return the value of the sum of the numbers from m to n
 */
public static int sum(int m, int n) {
    int val = 0;
    if (m == n) {
        val = n;
    }
    else {
        val = m + sum(m + 1, n);
    }
    return val;
}
```