1. C)
2. C)
3. D)

2.4. C)

2.5. B)

2.6. A)

2.7. E)

3.1. E)

3.2. I)

3.3. A)

3.4. J)

4.1. 8

4.2. D)

4.3. g(n) = O(f(n)). This is because f(n) grows faster than g(n) because its dominant term is to the power of 2 as compared to g(n)’s dominant term of 1.

4.4. f(n) = O(g(n)). This is because g(n) grows faster than f(n) because its dominant term is to the power of 2 as compared to f(n)’s dominant term of 1.

4.5. C)

4.6. C)

4.7. A)

4.8. A)

4.9. C)

5.a. This algorithm returns the sum of the squares of all integers from 0 to n.

5.b. s = s + 1\*1;

5.c. It’s executed n+1 times.

5.d. The efficiency class of this algorithm is O(n) or linear time.

5.e. The only better algorithm complexity possible is O(1). Since we know that we’re trying to do a mathematical sum, we can look for a formula that can give us the result without iteration. This gives us [n(n+1)(2n+1)]/6. We only have to compute this formula once and return its result, thus giving us an O(1) complexity algorithm.