1. For the sequences listed below.
   a) Find the best Big O for each sequence.
   b) List the sequences from slowest growth rate to fastest growth rate.

\[ 5\sqrt{n^7}, \quad n^2 \log(n^2), \quad 7 \log(n^7), \quad 0.1 n^5, \quad \left(\frac{9}{8}\right)^n, \quad \frac{1}{2} n^3 \log(n), \quad 6 \log(n) + n + n^2, \quad \frac{n^8 + 3n^2}{n + 2n^5}, \quad \log(783) \]

For each of the following code sections, find the best Big O for the growth rate (for time) in terms of n.

2. 
   ```
   for (i = 0; i < n; ++i )
      x += 3;
   for (j = 0; j < n; ++j )
      x *= 4;
   ```

3. 
   ```
   for (i = 0; i < n; ++i )
      for (j = 0; j < n; ++j )
         x += 2;
   ```

4. 
   ```
   for (i = 0; i < n; i += 1/n )
      x += 3;
   ```

5. 
   ```
   for (i = 1; i < n; i*=2 )
      x += 2;
   ```

6. 
   ```
   for (i = 1; i < n; i+=3 )
      x *= 7;
   ```

7. 
   ```
   for (i = 1; i < n; ++i )
      for (j = n; j > 1; j/=5 )
         ++x;
   ```

8. 
   ```
   k = 1
   for ( i = 1; i < n; ++i )
      if ( i%2 ) // if i is odd
         for ( j = 0; j < 3*n; ++j )
            cout << ++j;
      else // if i is even
         while ( k < i )
            cout << ++k;
   ```

9. Sort the array of n values
   ```
   for ( i = 0; i < m; ++i )
      binary search array1 for element i in array2
   ```

10. ```
    for ( i = 0; i < m; ++i )
        linear search array1 (which contains n elements) for element i in array2
    ```

11. When is the algorithm in #9 better (faster algorithm/has a slower growth rate) than the algorithm in #10?