1. What is the time complexity of the following code fragments, with respect to n? Give your answer in Θ-notation.

   (a)
   ```
   k = 1;
   for (i = 0; i < n; i++)
       k *= 2;
   ```

   (b)
   ```
   k = 1;
   for (i = 0; i < n; i++)
       temp = k;
       for (j = 0; j < k; j++)
           temp++;
       k = temp;
   ```

   (c)
   ```
   k = 1, x = 1;
   for (i = 0; i < n; i++)
       for (j = 0; j < x; j++)
           k++;
       x *= 2;
   ```

2. Order the following functions from smallest to largest asymptotic complexity. Identity any pairs of functions that have the same complexity (i.e. are Θ of each other).

   (a) 2^x
   (b) $\log x^2$
   (c) $x!$
   (d) $\sqrt{x}!$
   (e) $\log\left(\sqrt{x^{\log x}}\right)$
   (f) $\log^x x$
   (g) $\sum_{i=1}^{x} i$
   (h) $\sum_{i=1}^{x} x$

3. Consider the following two algorithms for sorting an array A of n comparable values $A[1], A[2], \ldots A[n]$.

   `Selection-Sort`

   for $i = 1 \to n-1$ **do**
     $jMin = i$
     for $j = i \to n$ **do**
       if $A[j] < A[jMin]$ **then**
         $jMin = j$
     swap $A[i]$ and $A[jMin]$

   `Insertion-Sort`

   for $i = 2 \to n$ **do**
     for $j = i \to 2$ **do**
       if $A[j] < A[j-1]$ **then**
         swap $A[j]$ and $A[j-1]$
       else
         break

   (a) Find the best-case and worst-case runtimes of `Selection-Sort` and `Insertion Sort`. Your answers should be in Θ notation.

   (b) True or False: In practice, `Insertion-Sort` will always run at least as fast as `Selection-Sort`.

(c) True or False: The worst-case runtime of `Selection-Sort` is $\Omega(n)$.

(d) True or False: The worst-case runtime of `Insertion-Sort` is $\Omega(n^2)$.

4. Given two sets A and B and their sizes m and n, the following algorithm calculates the sets' intersection (i.e. the elements they have in common). The sets are represented as sorted arrays.

```
intersect(A, m, B, n):
    C = new empty set
    for (i = 0; i < m; i++)
        for (j = 0; j < n; j++)
            if (A[i] == B[j])
                C.append(A[i])
                break
    return C
```

(a) What is the runtime complexity of this algorithm? Give your answer in Θ-notation in terms of m and n. (The `append` operation is constant-time.)

(b) Write an algorithm for the same task with a better runtime complexity.