1. **Rates of growth.** For each pair of functions, \( f(n) \) and \( g(n) \), place an “X” in the appropriate box indicating if \( f = O(g) \) or \( g = O(f) \). For each pair, you may mark one, both, or none of the boxes.

<table>
<thead>
<tr>
<th>( f(n) )</th>
<th>( g(n) )</th>
<th>( f = O(g) )</th>
<th>( g = O(f) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n(\log_2 n)^2 )</td>
<td>( n^2(\log_2 n) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n^2 )</td>
<td>( n \log_2 n )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n(\log_2 n)^4 )</td>
<td>( n^{1.2} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 200n^2 + n^{1.5} )</td>
<td>( n^2/500 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \log_7 n )</td>
<td>( \log_5 n )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n/\log_2 n )</td>
<td>( \sqrt{n} \log_2 n )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5^n )</td>
<td>( 7^n )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 7^n )</td>
<td>( 5^{(n^2)} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n! )</td>
<td>( (n + 1)! )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \sqrt{n} )</td>
<td>( (1 + (-1)^n)n )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. **Estimating sums by integrals.** Using the method of estimating a sum by an integral, show that

(a) \( \sum_{i=1}^{n} \ln(i) = n \ln(n) + O(n) \)
(b) \( \sum_{i=1}^{n} i \ln(i) = \frac{1}{2} n^2 \ln(n) + O(n^2) \)

**Note:** you can use an online calculator (such as Wolfram Alpha) to compute an indefinite integral.

3. **Limit ratio test.** Use the limit ratio test to show that the infinite series \( \sum_{i=1}^{\infty} i^2/2^i \) converges.

4. **Integral test.** For an infinite sum \( \sum_{i=k}^{\infty} a_i \), we can test for convergence by estimating the finite sum \( \sum_{i=k}^{\ell} a_i \) by an integral, and use this estimate to determine if the limit \( \lim_{\ell \to \infty} \sum_{i=k}^{\ell} a_i \) is finite or infinite. Use this technique to determine the convergence of the following series:

(a) \( \sum_{i=2}^{\infty} 1/(i \ln(i)) \)
(b) \( \sum_{i=2}^{\infty} 1/(i(\ln(i))^2) \)

**Note:** the limit ratio test is inconclusive on these examples; as above, you can use an online calculator to compute an indefinite integral.

5. **Mystery algorithm.** Consider the following algorithm, which operates on an array \( A[1..n] \) of integers.

```plaintext
for i in [1..n] do
    A[i] ← 0
for i in [1..n] do
    j ← i
    while j ≤ n do
        j ← j + i
```

(a) Show that the running time of this algorithm is \( O(n \log n) \).
(b) Describe in words the value of \( A[i] \) at the end of execution.
6. **Number of internal nodes in a 2-3 tree.** Prove that a 2-3 tree with \( n \) leaves has at most \( n - 1 \) internal nodes.

Prove this by induction on the height of the tree; that is, the induction hypothesis is:

\[
P(h) : \text{any 2-3 tree of height } h \text{ with } n \text{ leaves has at most } n - 1 \text{ internal nodes.}
\]

Prove \( P(0) \) directly (this is really trivial), and then prove that \( P(h) \implies P(h + 1) \) for all \( h \geq 0 \).

7. **Fibonacci numbers.** Recall the Fibonacci numbers: \( F_0 = 0, F_1 = 1, \) and \( F_{k+2} = F_k + F_{k+1} \).

(a) Prove that for all \( k \geq 0 \):

\[
F_{k+2} = 1 + \sum_{i=0}^{k} F_i.
\]

(b) Prove that for all \( k \geq 0 \):

\[
F_{k+2} \leq \phi^{k+1}, \text{ where } \phi = (1 + \sqrt{5})/2 \approx 1.618 \text{ is a root of } x^2 = x + 1.
\]

(c) Prove that for all \( k \geq 0 \):

\[
F_{k+2} \geq \phi^k, \text{ where } \phi \text{ is as in part (b)}.
\]

Prove these by induction on \( k \). For each part, formulate an appropriate induction hypothesis \( P(k) \). For parts (b) and (c), use so-called “strong induction” to prove this, by proving the following statement holds for each \( k \geq 0 \):

\[
P(0), P(1), \ldots, P(k-1) \implies P(k).
\]

Note that when \( k = 0 \), this statement is logically equivalent to saying that \( P(0) \) is true. However, for \( k > 0 \), you prove the statement by assuming \( P(0), \ldots, P(k-1) \) are true, and from this, showing that \( P(k) \) must be true as well. Contrast this to “ordinary induction”, where you have to prove that \( P(k) \) is true assuming only that \( P(k-1) \) is true.

8. **Dynamic vectors.** Consider the following implementation of a dynamic-sized vector class in Java:

```java
class Vector {
    int size;
    int capacity;
    double[] data;

    Vector () { size = 0; capacity = 1; data = new double[1]; }

    void resize(int newsz) {
        if (capacity < newsz) {
            capacity = newsz;
            double[] newdata = new double[capacity];
            for (int i = 0; i < size; i++)
                newdata[i] = data[i];
            data = newdata;
        }
        size = newsz;
    }

    void append(double val) { resize(size+1); data[size] = val; }
}
```

(a) Suppose that the following code is executed:

```java
Vector vec;
for (int i = 0; i < n; i++) vec.append(1.0);
```

Show that the number of times line 13 is executed is \( \Theta(n^2) \). The best way to do this is to give an exact formula for the number of times it is executed.

(b) Suppose that we modify line 10 so that it reads as follows:

```java
capacity = Math.max(2*capacity, newsz);
```
Suppose that we again execute the code in part (a). Show that the number of times line 13 is executed is $O(n)$. Again, the best way to do this is to give an exact formula for the number of times it is executed.

**Discussion.** Many programming languages provide such a dynamic-sized vector as a part of a standard library. As you can see, the strategy of doubling the capacity gives much better overall performance than the naive strategy of increasing it by the minimum required. With this strategy, the time required to perform $n$ append operations is $O(n)$. Because of this, it is sometimes said that the “amortized cost” of this strategy is $O(1)$, i.e., constant, which is to say that even though some append operations cost more and some cost less, averaging over a long sequence of such operations, each one takes a constant amount of time.