1. MERGE-SORT is an example of a *divide-and conquer* algorithm. The algorithm sorts an input sequence S of numbers along the following steps:

   - *divide*: split S into two parts of roughly equal length,
   - *conquer*: sort the subsequences recursively,
   - *combine*: merge the two sorted subsequences to produce the sorted output.

   In pseudocode:

   Algorithm 1: MERGE-SORT(S, p, r)

   ```
   if p < r then
       q ← ⌊(p + r)/2⌋
       MERGE-SORT(S, p, q)
       MERGE-SORT(S, q + 1, r)
       MERGE(S, p, q, r)
   ```

   (a) Explain the role of p and r in the algorithm specification. What values should they have upon initial invocation of the algorithm?

   (b) Write MERGE. It was claimed in class that MERGE can be implemented to run in $\Theta(r - p + 1)$ time. How does your implementation compare?

   (c) Implement MERGE-SORT in a language of your choice.
      i. Write a driver that invokes it with $S = (5, 2, 4, 6, 1, 3)$.
      ii. Build a container with 10^6 random numbers. Sort it using your implementation.

2. Consider the function Li_2 defined for $|z| \leq 1$ by

$$\mathrm{Li}_2(z) := -\sum_{n=1}^{\infty} \frac{z^n}{n} = -z - \frac{z^2}{2} - \frac{z^3}{3} - \frac{z^4}{4} - \cdots$$

   (a) In the language of your choice, implement a procedure $\texttt{dilog}(z, n)$ that computes the sum of the first n terms of the series for the given floating point number z.

   (b) Implement a procedure $\texttt{sdilog}(Z, n)$ that accepts a sequence Z of floating point numbers and computes the sum

$$\texttt{sdilog}(Z, n) := \sum_{z \in Z} \texttt{dilog}(z, n)$$

   (c) Build a cost model for $\texttt{sdilog}$ as a function of n and $m := \mathrm{length}(Z)$. Assume that additions and subtractions cost c_+ each, multiplications and divisions cost $c_\times$ each, and that raising a number to the nth power costs $c_\star$.