3. Write a function to parse any valid json string into a corresponding Object, List, or Map object. You can use C, C++, Java, Scala, Kotlin, Python, Node. Note that the integer and floating point should be arbitrary precision.

Answer:

For arbitrary precision, Java provides **BigInteger** for integers and **BigDecimal** for floating-point numbers.

Steps:

- Parse the JSON string using the org.json library.
- When encountering numbers, use BigDecimal for both integers and floating-point values to preserve precision.
- Recursively convert JSONObject and JSONArray into corresponding Map<String, Object> and List<Object> structures.

Key Concepts:

1. **BigDecimal for Arbitrary Precision**:
    - Both integers and floating-point numbers are converted to BigDecimal to ensure arbitrary precision.
2. **Recursive Parsing**:
    - The parseJSONObject method converts a JSONObject into a Map<String, Object>.
    - The parseJSONArray method converts a JSONArray into a List<Object>.
    - parseValue handles the recursive conversion of nested objects.
3. **convertPrimitive**:
    - This function converts any numeric value (Integer, Long, Float, Double) to BigDecimal.
    - For other primitive types (e.g., String, Boolean), the value is returned as is.