This report details the implementation of a Longest Path algorithm using the MapReduce programming model, enhanced with multithreading in Python

Key Components and Process:

- **Objective:** To implement a distributed MapReduce framework utilizing multithreading to efficiently find the file path with the greatest length from multiple data sources.
- **MapReduce Architecture:** This parallel, distributed model is used for processing large datasets.
    - **Mapper Function:** Takes an input line (a file path), calculates its length, and returns a key-value pair: `("longest", (length, path))`.
    - **Reducer Function:** Receives a list of `(length, path)` tuples associated with the key "longest" and uses the `max()` function to identify and return the tuple containing the maximum length and the corresponding longest path.
- **Implementation Details (Multithreading):**
    - **Map Task (`map_task`):** Processes a single input file in a separate thread. It reads the file line by line, calls the `mapper` for each line, and uses a `Lock` for thread-safe printing of the results before returning all key-value pairs from that file.
    - **Main Processing Flow:**
        - **MAP Phase (Parallel):** A `ThreadPoolExecutor` is used to manage up to 8 concurrent threads, each running a `map_task` on an input file. The results from all tasks are collected into `map_output`.
        - **SHUFFLE Phase:** The collected `map_output` is grouped by key (which is "longest") into a dictionary (`shuffled`), resulting in a single list of all `(length, path)` values.
        - **REDUCE Phase:** The `reducer` function is called with the key and the list of values to find and print the final maximum path length and the longest path.

Conclusion: The program successfully combines MapReduce with multithreading, specifically using `ThreadPoolExecutor` and `Lock`, to concurrently process multiple files and solve the longest path problem while ensuring thread safety.