1. Reading and Writing Text Files

- **Problem**: Write a program to read from one text file and write its contents to another file.
- **Example**: Given a text file `input.txt`, create a new file `output.txt` with the same content.
- **Objective**: Read from a file, process the data if needed, and write it to a new file.

2. Count Word Frequency in a Text File

- **Problem**: Write a Python program to read a text file and calculate the frequency of each word in the file.
- **Example**: Given a text file, count how many times each word appears and display the results in descending order of frequency.
- **Objective**: Read file contents, split the text into words, and count word frequencies using dictionaries.

3. Merging Multiple Text Files into One

- **Problem**: Merge the contents of multiple text files into a single file.
- **Example**: Given three text files (`file1.txt`, `file2.txt`, `file3.txt`), merge their contents into a new file `merged.txt`.
- **Objective**: Open and read multiple files, then write their contents into a single file.

4. File Comparison

- **Problem**: Write a Python program to compare two text files and highlight the differences between them.
- **Example**: Compare `file1.txt` and `file2.txt` line by line and print the differences.
- **Objective**: Read both files, compare their contents, and identify the differences (line by line or word by word).

5. Log File Analyzer

- **Problem**: Write a program to analyze a log file and extract useful information such as error counts and timestamps.
- **Example**: Given a server log file, count the number of error occurrences and list the corresponding timestamps.
- **Objective**: Parse and process log files to extract patterns like error messages, IP addresses, or other critical information.

6. CSV File Reader and Writer

- **Problem**: Write a program to read data from a CSV file, modify the data, and write it to another CSV file.
- **Example**: Given a CSV file containing user information, modify one of the fields (e.g., change email addresses) and save the changes to a new CSV file.
- **Objective**: Work with CSV files, reading and writing data using Python's `csv` module.

7. Parsing JSON Files

- **Problem**: Write a program to read a JSON file, parse the data, and print it in a human-readable format.
- **Example**: Given a JSON file with nested structures, extract specific pieces of data (e.g., user details) and display them.
- **Objective**: Read and parse JSON files using Python's `json` module and extract specific data fields.

8. Binary File Reader and Writer

- **Problem**: Write a program to read binary data from a file, manipulate it, and write the modified data back to a new file.
- **Example**: Read a binary image file, modify some of the pixel values, and save it as a new image.
- **Objective**: Work with binary files using Python's file I/O methods like `rb` and `wb` modes.

9. File Encryption and Decryption

- **Problem**: Implement a program that encrypts the content of a text file and decrypts it back to its original form.
- **Example**: Given a text file, use a simple encryption algorithm (like Caesar Cipher or XOR) to encrypt the file content and then decrypt it.
- **Objective**: Read from a file, encrypt the data, write it to a new file, and implement the reverse process to decrypt.

10. File Compression and Decompression

- **Problem**: Write a program to compress a file using gzip or zip and decompress it back to its original form.
- **Example**: Compress a text file `data.txt` into `data.zip` and then decompress it.
- **Objective**: Work with file compression and decompression using Python's `gzip` or `zipfile` module.

11. Counting Lines, Words, and Characters in a File

- **Problem**: Write a program to count the number of lines, words, and characters in a text file.
- **Example**: Given a text file, calculate the number of lines, words, and characters, and print the results.
- **Objective**: Read the file and perform basic text analysis to count lines, words, and characters.

12. Finding Duplicate Files in a Directory

- **Problem**: Write a program to scan a directory and identify duplicate files based on file content (not file names).
- **Example**: Compare all files in a given directory and identify any files that have the same content.
- **Objective**: Use hashing or byte-by-byte comparison to identify duplicate files in a directory.

13. Directory Walker (Recursive File Listing)

- **Problem**: Write a Python program to recursively list all files and directories in a given directory.
- **Example**: Given a directory, output a tree-like structure of all files and subdirectories within it.
- **Objective**: Use `os` or `os.path` to traverse directories and print their structure.

14. Renaming Files in a Directory

- **Problem**: Write a program to rename all files in a directory according to a specific pattern.
- **Example**: Rename all `.txt` files in a directory by adding a prefix like `old_` to their names.
- **Objective**: Use Python's `os` or `os.rename()` function to batch rename files based on specific criteria.

15. File Metadata Extraction

- **Problem**: Write a program to extract and display metadata (such as file size, creation date, and modification date) of files in a directory.
- **Example**: Given a directory of files, print each file's size, creation date, and last modification date.
- **Objective**: Use Python's `os` or `os.stat()` methods to retrieve and display file metadata.

16. Log File Rotation

- **Problem**: Implement a program that simulates **log file rotation** by renaming old log files and creating a new log file when the current one reaches a certain size.
- **Example**: When the current log file exceeds 1MB, rename it and start a new log file.
- **Objective**: Work with file sizes, renaming, and writing to new files to implement file rotation behavior.

17. Finding and Replacing Text in Files

- **Problem**: Write a program that reads a file, searches for a specific text pattern, and replaces it with another text.
- **Example**: Search for the word "error" in a log file and replace it with "warning."
- **Objective**: Use Python's file reading and writing capabilities to perform search-and-replace operations in text files.

18. Generating and Writing Large Data Files

- **Problem**: Write a program to generate large data files, such as files containing random numbers, text, or CSV data.
- **Example**: Generate a file with 10 million random integers and write it to a file.
- **Objective**: Use Python to generate large datasets and write them efficiently to disk.

19. Checksum Calculation for Files

- **Problem**: Write a program to calculate a file's checksum (e.g., MD5 or SHA256) to ensure its integrity.
- **Example**: Compute the MD5 hash of a file and verify its integrity by comparing it to a known hash.
- **Objective**: Use Python's `hashlib` module to compute and verify file checksums.

20. File Permissions Checker

- **Problem**: Write a Python program that checks the permissions of files in a directory and identifies files with insecure permissions.
- **Example**: Scan a directory and identify files that are world-writable or have other risky permissions.
- **Objective**: Use Python's `os` and `stat` modules to check file permissions and highlight any potential security risks.