Task 1: Easy

Create and List Files Using Wildcards

1. Create the following files using a single command:
   - `file1.txt`
   - `file2.txt`
   - `file3.txt`
2. List all `.txt` files in the current directory using a wildcard pattern.

Solution:

```
# Create the files
touch file{1..3}.txt

# List all .txt files
ls *.txt
```

- `touch file{1..3}.txt` creates `file1.txt`, `file2.txt`, and `file3.txt`.
- `ls *.txt` lists all `.txt` files in the current directory.

Task 2: Medium

Filter and Copy Files

1. Create the following files:
   - `reportA.txt`
   - `reportB.txt`
   - `reportC.txt`
   - `summaryA.log`
   - `summaryB.log`
2. Copy only the `.txt` files starting with `report` to a new directory named `txt_files`.

Solution:

```
# Create the files
touch report{A,B,C}.txt summary{A,B}.log

# Create the target directory
mkdir -p txt_files

# Copy .txt files starting with "report"
cp report*.txt txt_files/
```

- `touch report{A,B,C}.txt` creates the `.txt` files.
- `mkdir -p txt_files` ensures the directory exists.
- `cp report*.txt txt_files/` copies only `.txt` files that start with `report`.

Task 3: Hard

Find and Process Files Recursively

1. Create the following directory structure with files:

```
project/
   src/
       main.cpp
       utils.h
   docs/
       guide.txt
       readme.md
   logs/
       app.log
       error.log
```

2. Using a single `find` command:

    - Locate all `.log` files in the `logs/` directory and display their content.
    - Exclude `error.log` from the output.

Solution:

```
# Create the directory structure and files
mkdir -p project/{src,docs,logs}
touch project/src/{main.cpp,utils.h}
touch project/docs/{guide.txt,readme.md}
touch project/logs/{app.log,error.log}

# Find and display content of .log files, excluding error.log
find project/logs/ -name "*.log" ! -name "error.log" -exec cat {} \;
```

- `mkdir -p` creates the directory structure.
- `touch` creates the specified files.
- `find project/logs/ -name "*.log" ! -name "error.log" -exec cat {} \;` finds all `.log` files except `error.log` and displays their content.

Task 4: Hard

Delete Specific Files Based on Patterns

1. Create the following files:
    - `temp_001.log`
    - `temp_002.log`
    - `temp_001.bak`

    - `temp_002.bak`
    - `temp_003.tmp`
2. Delete all `.log` and `.bak` files but leave `.tmp` files untouched.

Solution:

```
# Create the files
touch temp_001.log temp_002.log temp_001.bak temp_002.bak temp_003.tmp

# Delete .log and .bak files
rm temp_*.{log,bak}
```

- `touch` creates the specified files.
- `rm temp_*.{log,bak}` deletes all `.log` and `.bak` files using brace expansion.

Task 5: Hard

Find and Archive Files Modified in the Last 3 Days

1. Create the following directory structure with files:

```
archive_project/
   day1/
       file1.txt
       file2.txt
   day2/
       file3.log
       file4.log
   day3/
       file5.md
       file6.md
```

2. Archive (zip) all files modified in the last 3 days into a file named `recent_files.zip`.

Solution:

```
# Create the directory structure and files
mkdir -p archive_project/{day1,day2,day3}
touch archive_project/day1/{file1.txt,file2.txt}
touch archive_project/day2/{file3.log,file4.log}
touch archive_project/day3/{file5.md,file6.md}

# Find and archive files modified in the last 3 days
find archive_project/ -mtime -3 -type f -exec zip recent_files.zip {} +
```

- `mkdir -p` creates the directories.
- `touch` creates the files.

- `find ... -exec zip ...` finds the files and adds them to the zip archive.

Task 6: Hard

Organize Files by Type into Separate Directories

1. Create the following files in a single directory:
   - `image1.jpg`
   - `image2.png`
   - `document1.pdf`
   - `document2.docx`
   - `script1.sh`
   - `script2.py`
2. Organize these files into separate directories based on their extensions:
   - Move `.jpg` and `.png` files to an `images/` directory.
   - Move `.pdf` and `.docx` files to a `documents/` directory.
   - Move `.sh` and `.py` files to a `scripts/` directory.

Solution:

```
# Create the files
touch image1.jpg image2.png document1.pdf document2.docx script1.sh script2.py

# Create the target directories
mkdir -p images documents scripts

# Move files to their respective directories
mv *.jpg *.png images/
mv *.pdf *.docx documents/
mv *.sh *.py scripts/
```

- `touch` creates the files.
- `mkdir -p` ensures the directories exist.
- `mv` moves the files to their respective directories using wildcards.