File System Hierarchy Problem

File management systems, such as Windows Explorer or Finder on Macs, utilize data structures such as trees and lists to represent the hierarchy of the file system. Your task is to implement a program that models of files and folders as objects and uses data structures to manage the structure and traversal within the file system.

Your program should be able to perform the following operations:

- Create a new folder
- Create a new file
- Move a file or folder to a different location
- Delete a file or folder
- Print the current file system hierarchy

With the following input criteria:

- File or folder names will be unique and contain only alphanumeric characters.
  Create operation should fail if the supplied file name already exists.
- Move and delete operation should fail if the supplied file or folder name does not exist in current file system.
- Create operation should fail if the supplied parent folder name does not exist in current file system.
- It is guaranteed that there will be no circular references in the file system hierarchy.
- Input strings are case-insensitive.

Input

Each line of input contains a string representing the operation to perform and its corresponding parameters. The format of each line will be one of these followings:

- create_folder [folder_name] [parent_folder_name]?
- create_file [file_name] [parent_folder_name]?
- move [file_or_folder_name] [destination_folder_name]?
- delete [file_or_folder_name]
- print
- exit

Parameters `parent_folder_name` and `destination_folder_name` are optional. When not provided, the operation should assume the top level (root) of the file system.

The program should process input strings until the `exit` command is received.

Output

For the `print` operation, program should display current hierarchy of the file system, with each level indented by spaces or a tab. Each file or folder should be listed on a new line.

Example Input

```
create_folder dirA
create_folder dirB
create_file fileC dirB
create_file fileD dirA
create_file fileE
create_file fileF dirB
move dirB dirA
print
exit
```

Example Output

```
dirA
  fileD
  dirB
    fileC
    fileF
fileE
```

Marking criteria

This problem assesses your skills in object-oriented programming, knowledge of data structures, code readability, and extensibility. The goal is to test your understanding of the problem, ability to create models with their behavior, and the use of appropriate data structures to solve the problem.

Conditions

- You may use any programming language.
- Browsing internet and using AI tools are allowed.
- Completing the task within time limit is not required, but it should be done to the best of your ability.

Tips

- Start by focusing on the design of classes and interfaces. Once these are complete, work on implementation. Keep input parsing, displaying results, and other details at the last.
- Use meaningful names for classes, variables, and methods.
- Handling edge cases, such as ensuring input correctness and managing duplicated or missing file and folder names, is a plus.
- Helper methods such as `find()` or `print()` can be useful in various scenarios.
- If you have any doubts or questions, please ask for clarification.