Lab1 测试用例

测试用例 1：基本编辑功能

测试输入：

```Mathematica
load test1.md
insert ## 程序设计
append-head # 我的资源
append-tail ### 软件设计
append-tail #### 设计模式
append-tail 1. 观察者模式
append-tail 3. 单例模式
insert 6 2. 策略模式
delete 单例模式
append-tail 3. 组合模式
list-tree
append-tail ## 工具箱
append-tail ### Adobe
list-tree
save
```

期望输出：

有序列表是文本项，属于最近一个标题项的叶子节点。删除时只指定了标题名或者文本名，应该删除所有字符串等于“单例模式”的标题或者文本。因此第一次 `list-tree` 结果如下：

```Markdown
```

```
└── 我的资源
    └── 程序设计
        └── 软件设计
            └── 设计模式
                ├── 1. 观察者模式
                ├── 2. 策略模式
                └── 3. 组合模式
```

第二次 `list-tree` 的结果如下：

```Markdown
└── 我的资源
    ├── 程序设计
    │   └── 软件设计
    │       └── 设计模式
    │           ├── 1. 观察者模式
    │           ├── 2. 策略模式
    │           └── 3. 组合模式
    └── 工具箱
        └── Adobe
```

执行 `save` 指令后，用文本编辑器打开文件 test1.md，内容如下：

```Markdown
# 我的资源
## 程序设计
### 软件设计
#### 设计模式
1. 观察者模式
2. 策略模式
3. 组合模式
## 工具箱
### Adobe
```

测试用例 2：撤销与重做编辑操作

测试输入：

```Markdown
```

```
load test2.md
append-head # 旅行清单
append-tail ## 亚洲
append-tail 1. 中国
append-tail 2. 日本
delete 亚洲
undo
redo
list-tree
save
```

期望输出：

`undo` 应该撤销上一步 `delete` 操作，而 `redo` 会重做上一次 `undo` 撤销的 `delete` 命令，因此 `list-tree` 的结果如下：

Markdown

```
└── 旅行清单
    ├── 1. 中国
    └── 2. 日本
```

执行 `save` 指令后，用文本编辑器打开文件 test2.md，内容如下：

Markdown

```
# 旅行清单
1. 中国
2. 日本
```

测试用例 3：混合编辑操作的撤销与重做

测试输入：

Markdown

```
load test3.md
append-head # 书籍推荐
append-tail * 《深入理解计算机系统》
undo
append-tail ## 编程
append-tail * 《设计模式的艺术》
redo
list-tree
```

```
append-tail * 《云原生：运用容器、函数计算和数据构建下一代应用》
append-tail * 《深入理解Java虚拟机》
undo
redo
list-tree
save
```

期望输出：

由于 `redo` 的上一次编辑命令是 `append-tail * 《设计模式的艺术》` 不是 `undo`，因此 `redo` 不会重做上一次 `undo` 撤销的操作。第一次 `list-tree` 的结果如下：

Markdown

```
└── 书籍推荐
    └── 编程
        └── ·《设计模式的艺术》
```

第二次 `redo` 命令的上一个编辑命令是 `undo`，需要重做 `append-tail * 《深入理解Java虚拟机》`，因此第二次 `list-tree` 的结果如下：

Markdown

```
└── 书籍推荐
    └── 编程
        ├── ·《设计模式的艺术》
        ├── ·《云原生：运用容器、函数计算和数据构建下一代应用》
        └── ·《深入理解Java虚拟机》
```

执行 `save` 指令后，用文本编辑器打开文件 test3.md，内容如下：

Markdown

```
# 书籍推荐
## 编程
* 《设计模式的艺术》
* 《云原生：运用容器、函数计算和数据构建下一代应用》
* 《深入理解Java虚拟机》
```

测试用例 4： 切换工作区

测试输入：

```Markdown
load test4.md
append-head # 旅行清单
append-tail ## 亚洲
save
append-tail 1. 中国
append-tail 2. 日本
append-tail ## 欧洲
load test3.md
list-tree
load test4.md
list-tree
```

期望输出：

执行 `load` 操作加载文件后，可以多次使用 `save` 命令将内存中的数据进行持久化处理，未持久化处理的部分数据将丢失。以上编辑过程中执行了 `load test3.md` 指令会切换工作区至 test3.md，此时第一次 `list-tree` 显示 test3.md 里的内容：

```Markdown
└── 书籍推荐
    └── 编程
        ├── ·《设计模式的艺术》
        ├── ·《云原生：运用容器、函数计算和数据构建下一代应用》
        └── ·《深入理解Java虚拟机》
```

由于上一次执行 `load` 指令切换工作区前没有保存 test4.md 的部分内容，因此再次切换工作区至 test4.md 后，工作区为上一次 test4.md 保存后的内容。`append-tail 1. 中国` `append-tail 2. 日本` `append-tail ## 欧洲` 等操作后没有执行 save 指令，该部分数据不会保存到 test4.md 文件中：

```Markdown
└── 旅行清单
    └── 亚洲
```

测试用例 5：混合所有功能

测试输入：

```Markdown
load test5.md
append-head # 旅行清单
append-tail ## 欧洲
insert 2 ## 亚洲
insert 3 1. 中国
insert 4 2. 日本
save
undo
list-tree
delete 亚洲
list-tree
history 2
undo
list-tree
redo
list-tree
redo
list-tree
save
```

期望输出：

第一次 `undo` 的上一个命令是 `save`，不可以被跳过，因此 `undo` 不生效。第一次 `list-tree` 的内容如下：

```Markdown
└── 旅行清单
    ├── 亚洲
    │   ├── 1. 中国
    │   └── 2. 日本
    └── 欧洲
```

`delete` 指令只删除了 `亚洲` 所在的标题。第二次 `list-tree` 的内容如下：

```Markdown
└── 旅行清单
    ├── 1. 中国
```

```
    ├── 2. 日本
    └── 欧洲
```

第一次执行 `history 2`，显示最近执行的两条命令，及命令执行的时间戳。参考内容如下：

```Markdown
202310xx xx:xx:xx list-tree
202310xx xx:xx:xx delete 亚洲
```

第三次 `list-tree`，由于 `undo` 之前的 `history` 与 `list-tree` 命令属于显示命令组，应该被跳过，因此需要撤销 `delete 亚洲` 命令。

```Markdown
└── 旅行清单
    ├── 亚洲
    │   ├── 1. 中国
    │   └── 2. 日本
    └── 欧洲
```

第四次 `list-tree`，由于 `redo` 之前的 `list-tree` 命令属于显示命令组，应该被跳过，因此需要重做上一次 `undo` 撤销的命令，即重做 `delete 亚洲` 操作。

```Markdown
└── 旅行清单
    ├── 1. 中国
    ├── 2. 日本
    └── 欧洲
```

第五次 `list-tree`。每一个 `redo` 都要有与之对应的 `undo` 命令配对，上一次 `redo` 配对成功后，上一个编辑命令变为 `delete 亚洲`，再上一个编辑命令才是 `undo`，因此最新的 `redo` 没有与之配对的 `undo`， 不需要重做。

```Markdown
└── 旅行清单
    ├── 1. 中国
    ├── 2. 日本
    └── 欧洲
```

执行 `save` 指令后，用文本编辑器打开文件 test5.md，内容如下：

```Markdown
# 旅行清单
1. 中国
2. 日本
## 欧洲
```