

Lab1 测试用例

测试用例1：基本编辑功能

测试输入：

```
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：

```
├─ 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：

```
├─ 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：

```
# 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：

```
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：

```
└─ 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：

```
# 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：

```
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：

```
└─ 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：

```
└─ 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：

```
# 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：

```
load test4.md
append-head # 旅行清单
append-tail ## 亚洲
ws
save
append-tail 1. 中国
load test3.md
list-tree
ws
append-tail * 《软件工程》
switch 1
save
switch 2
close 2
n
ws
```

期望输出：

1-4行：新建`test4.md`后，追加两条内容，此时只加载了`test4.md`，且该文件正在编辑，且尚未保存，所以第四行的`ws`显示如下：

```
1 test4.md*<
```

5-8行：保存`test4.md`内容，并追加了新内容，然后在未保存新内容的情况下切换到`test3.md`，此时第8行的`list-tree`显示`test3.md`里的内容：

```
└─ 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》
```

第9行`ws`显示如下：

```
1 test4.md*
2 test3.md<
```

向`test3.md`追加内容 * 《软件工程》后，切换到第一个文件，并保存第一个文件`test4.md`的内容。接着切换到`test3.md`，然后关闭`test3.md`。由于该文件有尚未保存的修改，所以询问用户是否要保存文件(y/n)，用户选择了n，所以先前追加的* 《软件工程》被丢弃，提示输出如下：

```
save changed file before closing? (y/n)
```

此外，当前正在编辑的文件变成上一个序号的文件，即`test4.md`，所以第15行`ws`的显示如下：

```
1 test4.md<
```

测试用例 5：混合所有功能

测试输入：

```
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：

```
└─ 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：

```
└─ 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：

```
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。

```
└─ 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。

```
└─ 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。

```
└─ 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：

```
# 旅行清单
1. 中国
2. 日本
## 欧洲
```