Design/implementation

1. **Warmup**: extend the design of *memoryfs_shell.py* to support the following shell commands:

`mkdir dirname`

- calls FileName.create with type=INODE_TYPE_DIR

`create filename`

- calls FileName.create with type=INODE_TYPE_FILE

`append filename string`

- ensures filename is in cwd and for valid file
- Increases size by len(string) if under MAX_FILE_SIZE
- calls FileName.Write at offset (previous size)

2. **Main challenge**: extend your design to support unlinking (removing). In particular, you will extend the FileName() class in memoryfs.py with the Unlink() method, and extend the shell with the rm command (which calls Unlink()):

memoryfs_shell.py: `rm filename`

- calls FileName.Unlink

memoryfs.py `def Unlink(self, dir, name):` a) *InodeNumber.InodeNumberToInode()* to get dirinode b) *Lookup* c) *InodeNumber.InodeNumberToInode()* to get fileinode d) *InodeNumber.StoreInode()* e) Saved last filename entry. Linear searched blocks for filename entry and swapped f) *InodeNumber.StoreInode()* g) *InodeNumber.StoreInode()*

Testing:

- `create`: fails when no available inodes, able to view directory entry and inode
- `mkdir`: able to cd in/out created dir, able to view inode and create files inside
- `append`: ensured does not allow over max file size
- ensured working with different fsconfig using `python3 memoryfs_shell.py -bs 512 -nb 2048 -ni 128 -is 32`
- ensured block size didn't increase during unlink()
- ensured removing 1/9 files with default configuration, last file would replace the entry location
    - checked directory entries using showblockslice
    - checked inode refcnt and size decremented