Design/implementation

`def Link(self, target, name, cwd):`

- Check for errors
- used *GeneralPathToInodeNumber* to get access to target inode (or check if it exists)
- *InsertFilenameInodeNumber* to add entry (name, target_inode) to cwd
- Increment target and cwd refcnt

`lnh target name`

- calls Link and prints errors

`def Symlink(self, target, name, cwd):`

- Check for errors
- used *GeneralPathToInodeNumber* to get access to target inode (or check if it exists)
- Updated `def Create(self, dir, name, type):` to support `INODE_TYPE_SYMLINK` and called to create sym_inode
- Allocate data block and store target string
- *InsertFilenameInodeNumber* to add entry (name, sym_inode) to cwd

`lns target name`

- calls Symlink and prints errors

`def ls`

- updated to check if type == `INODE_TYPE_SYM` then append @ and print target (first block of sym_inode)

Testing

- Ran example on assignment page
- Ensured Link incremented both refcnts and lns incremented cwd refcnt
- Ensured functions returned tuples (errorcode, error)
- Ensured Link created entry to same inode
    - Modifying contents in target to see if Linked file changes
    - using `showblockslice` to confirm inode number is the same for both link and target entries
- Ensured Symlink properly created new inode
    - Ensured correct size
    - Ensured correct target was added to block
- Ensured ls properly appended when listing symlinks
- Tested with absolute paths