1. **Linky Paths** Implement `linky_paths` which takes in a `Tree t` and modifies each label to be the path from that node to the root.

```python
def linky_paths(t):
    """
    >>> t = Tree(1, [Tree(2)])
    >>> linky_paths(t)
    >>> t
    Tree(Link(1), [Tree(Link(2, Link(1)))]
    """
    def helper(t, path_so_far):
        t.label = ___________________________________
        for __________________________________________:
            __________________________________________
        helper(___________________, ___________________)
```
2. **Find File Path** Implement `find_file_path` which takes in a `Tree t` and a string `file_str` and returns the full path of a file that we search for if the file exists. If the file does not exist, then return `None`.

```python
def find_file_path(t, file_str):
    ""
    >>> t = Tree('data', [Tree('comm', [Tree('dummy.py')]),
                        Tree('ecc',
                        [Tree('hello.py'), Tree('file.py')]), Tree('file2.py')])
    >>> find_file_path(t, 'file2.py')
    '/data/file2.py'
    >>> find_file_path(t, 'dummy.py')
    '/data/comm/dummy.py'
    >>> find_file_path(t, 'hello.py')
    '/data/ecc/hello.py'
    >>> find_file_path(t, 'file.py')
    '/data/ecc/file.py'
    """
    def helper(t, file_str, path_so_far):
        if __________________________________________________:
            return __________________________________________
        elif t.is_leaf():
            return
        for _________________________________________________:
            result = _________________________________________
            if _______________________________________________
                return result
        return ______________________________________________
```
2 Linked Lists

1. **Convert to String** Implement `convert_to_string` which takes in a Linked List `link` and converts the Linked List to a file path.
   
   ```python
def convert_to_string(link):
    
    >>> link = Link( data , Link( file2.py ))
    >>> convert_to_string(link)
    '/data/file2.py'
    
    if ________________:
        
        return ________________
    
    return ________________
```

2. **All Paths Linked** Implement `all_paths_linked` which takes in a Tree `t` and returns a list of all paths from root to leaf in a tree with one catch – each path is represented as a linked list.

   ```python
def all_paths_linked(t):
    
    >>> t1 = Tree(1, [Tree(2), Tree(3)])
    >>> t2 = Tree(1, [Tree(2), Tree(3, [Tree(4), Tree(5)])])
    >>> all_paths(t1)
    [Link(1, Link(2)), Link(1, Link(3))]
    >>> all_paths(t2)
    [Link(1, Link(2)), Link(1, Link(3, Link(4))), Link(1, Link(3, Link(5)))]
    
    if ________________:
        
        return ________________
    
    result = []
    for branch in t.branches:
        result = ________________
    
    return result
```
3. **Find File Path 2** Implement `find_file_path` which takes in a Tree `t` and a string `file_str` and returns the full path of a file that we search for if the file exists. If the file does not exist, then return `None`.

For this question, use the definition of `all_paths_linked` and `convert_to_string`.

```python
def find_file_path2(t, file_str):
    """
    >>> t = Tree('data', [Tree('comm', [Tree('dummy.py')]),
                       Tree('ecc',
                            [Tree('hello.py'), Tree('file.py')]),
                       Tree('file2.py')])
    >>> find_file_path2(t, 'file2.py')
    '/data/file2.py'
    >>> find_file_path2(t, 'dummy.py')
    '/data/comm/dummy.py'
    >>> find_file_path2(t, 'hello.py')
    '/data/ecc/hello.py'
    >>> find_file_path2(t, 'file.py')
    '/data/ecc/file.py'
    """
    for link in ________________________________:
        original = ________________________________

        while ________________________________:
            if ________________________________:
                return ________________________________
```

4. Skip Implement `skip` which takes in a Linked List `lnk` and an integer `n` which is great than 1 and mutates `lnk` such that every `n`th element is skipped.

```python
def skip(lnk, n):
    ""
    >>> lnk = Link(1, Link(2, Link(3, Link(4, Link(5, Link(6)))))
    >>> skip(lnk, 2)
    >>> lnk
    Link(1, Link(3, Link(5)))
    >>> lnk2 = Link(1, Link(2, Link(3, Link(4, Link(5, Link(6))))))
    >>> skip(lnk2, 4)
    >>> lnk2
    Link(1, Link(2, Link(3, Link(5, Link(6)))))
    ""
```

```python
def skipper(lst):
    count += 1
    if ________________________________:
        return
    else:
        lst.rest = ______________________
        count = __________________________
```

CS61A Exam Prep Spring 2018: Shea Conlon, Nipun Ramakrishnan and Tiffany Perumpail