Linked Lists and Midterm Review

Computer Science Mentors CS 61A

March 18 to March 20, 2019

1 Linked Lists

For each of the following problems, assume linked lists are defined as follows:

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
```

To check if a `Link` is empty, compare it against the class attribute `Link.empty`:

```
if link is Link.empty:
    print('This linked list is empty!')
```

1. What will Python output? Draw box-and-pointer diagrams to help determine this.

```
>>> a = Link(1, Link(2, Link(3)))
>>> a.first

>>> a.first = 5
>>> a.first

>>> a.rest.first

>>> a.rest.rest.rest.rest.first


>>> a.rest.rest.rest = a
>>> a.rest.rest.rest.rest.first
```

CS61A Spring 2019: Catherine Cang and Alex Stennet, with
Albert Xu, Albert Yu, Alex Yang, Allan Yu, Amy Hung, Chris Niu, Danelle Nachum, Evan Corriere, Jack Coyle, Jade Singh, Jason Chang, Jemmy Zhou, Kevin Wang, Lucas Pan, Mary Xu, Michelle Cheung, Olivia Shiah, Patricia Ouyang, Richard Roggenkemper, Samy Cherfaoui, Shreya Sahoo, Varun Jhunjhunwalla

2. Write a function `skip`, which takes in a `Link` and returns a new `Link` with every other element skipped.

```
def skip(lst):
    """
    >>> a = Link(1, Link(2, Link(3, Link(4))))
    >>> a
    Link(1, Link(2, Link(3, Link(4))))
    >>> b = skip(a)
    >>> b
    Link(1, Link(3))
    >>> a
    Link(1, Link(2, Link(3, Link(4)))) # Original is unchanged
    """
    if ____________________________________________:

        ____________________________________________:

    elif __________________________________________:

        ____________________________________________

    ________________________________________________
```

3. Now write function `skip` by mutating the original list, instead of returning a new list. Do NOT call the `Link` constructor.

```
def skip(lst):
    """
    >>> a = Link(1, Link(2, Link(3, Link(4))))
    >>> skip(a)
    >>> a
    Link(1, Link(3))
    """
```

4. Write a function `reverse`, which takes in a `Link` and returns a new `Link` that has the order of the contents reversed.
   Hint: You may want to use a helper function if you're solving this recursively.

```
def reverse(lst):
    """
    >>> a = Link(1, Link(2, Link(3)))
    >>> b = reverse(a)
    >>> b
    Link(3, Link(2, Link(1)))
    >>> a
    Link(1, Link(2, Link(3)))
    """
```

2 Midterm Review

For each of the following problems, assume the Tree class is defined as follows: For the

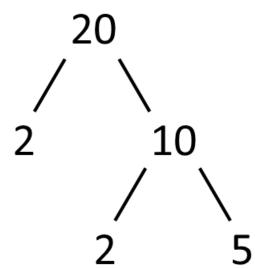
following problems, use this definition for the Tree class:

```
class Tree:
    def __init__(self, label, branches=[]):
        self.label = label
        self.branches = list(branches)

    def is_leaf(self):
        return not self.branches

    def __repr__(self):
        if self.branches:
            branch_str = ', ' + repr(self.branches)
        else:
            branch_str = ''
        return 'Tree({0}{1})'.format(self.label, branch_str)
```

1. Write a function that returns true only if there exists a path from root to leaf that contains at least `n` instances of `elem` in a tree `t`.

```
def contains_n(elem, n, t):
    """
    >>> t1 = Tree(1, [Tree(1, [Tree(2)])])
    >>> contains(1, 2, t1)
    True
    >>> contains(2, 2, t1)
    False
    >>> contains(2, 1, t1)
    True
    >>> t2 = Tree(1, [Tree(2), Tree(1, [Tree(1), Tree(2)])])
    >>> contains(1, 3, t2)
    True
    >>> contains(2, 2, t2) # Not on a path
    False
    """
    if n == 0:

        return True

    elif ________________________________________________:

        return __________________________________________

    elif ________________________________________________:

        return __________________________________________

    else:

        return __________________________________________
```

2. Define the function `factor_tree` which takes in a positive integer `n` and returns a factor tree for `n`. In a factor tree, multiplying the leaves together is the prime factorization of the root, `n`. See below for an example of a factor tree for `n = 20`.




```
def factor_tree(n):
    """
    >>> factor_tree(20)
    Tree(20, [Tree(2), Tree(10, [Tree(2), Tree(5)])])
    >>> factor_tree(1)
    Tree(1)

    for i in ______________________:

        if _________________________:

            return Tree(_____, _______________________________)

    __________________________________
```

CS61A Spring 2019: Catherine Cang and Alex Stennet, with
Albert Xu, Albert Yu, Alex Yang, Allan Yu, Amy Hung, Chris Niu, Danelle Nachum, Evan Corriere, Jack Coyle, Jade Singh, Jason Chang, Jemmy Zhou, Kevin Wang, Lucas Pan, Mary Xu, Michelle Cheung, Olivia Shiah, Patricia Ouyang, Richard Roggenkemper, Samy Cherfaoui, Shreya Sahoo, Varun Jhunjhunwalla

3. Draw the environment diagram that results from running the following code. If the code errors, draw the environment diagram up to the point that the error occurs.

```
earth = [0]
earth.append([earth])

def wind(fire, groove):
    fire[1][0][0] = groove
    def fire():
        nonlocal fire
        fire = lambda fantasy: earth.pop(1).extend(fantasy)
        return fire(groove)
    return fire()

sep = earth[1]
wind(earth, [earth[0]] + [earth.append(0)])
```