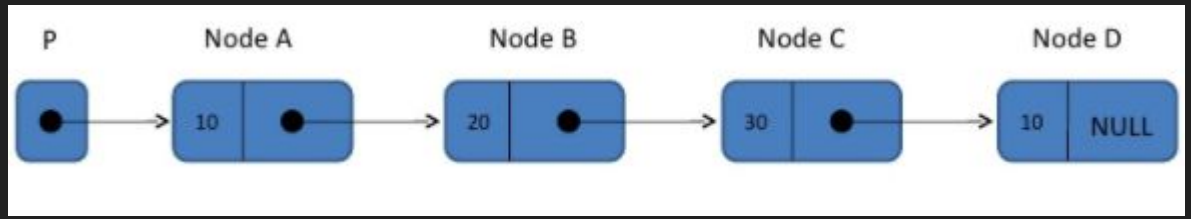
CS61A Discussion 6

Linked Lists and MT Review

Linked Lists

- New Abstract Data Structure!
  - List made up of nodes, each with a value (first) and a next
  - First: Any object
  - Next: Must be another LinkedList Instance
- Kind of like trees restricted to one branch



```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        self.first = first
        self.rest = rest
    def __getitem__(self, i):
        if i == 0:
            return self.first
        return self.rest[i-1]
    def __len__(self):
        return 1 + len(self.rest)
    def __repr__(self):
        if self.rest is Link.empty:
            return 'Link({})'.format(self.first)
        else:
            return 'Link({}, {})'.format(self.first,
                                         repr(self.rest))
```

Double Linked List Example

- https://goo.gl/jCb7Y5

Worksheet Time (1.1, 1.3)

Reverse Intuition

- Really needs leap of faith
- Reverse position of first node and assume our recursive call works perfectly!
- Sometimes tracking the progression through the recursive stack works, but in this case easier to use leap of faith

Worksheet (Widest Level)

(d) (6 pt) Implement `double_up`, which mutates a linked list by inserting elements so that each element is adjacent to an equal element. The `double_up` function inserts as few elements as possible and returns the number of insertions. The `Link` class appears on the midterm 2 study guide.

```
def double_up(s):
    """Mutate s by inserting elements so that each element is next to an equal.

    >>> s = Link(3, Link(4))
    >>> double_up(s) # Inserts 3 and 4
    2
    >>> s
    Link(3, Link(3, Link(4, Link(4))))
    >>> t = Link(3, Link(4, Link(4, Link(5))))
    >>> double_up(t) # Inserts 3 and 5
    2
    >>> t
    Link(3, Link(3, Link(4, Link(4, Link(5, Link(5))))))
    >>> u = Link(3, Link(4, Link(3)))
    >>> double_up(u) # Inserts 3, 4, and 3
    3
    >>> u
    Link(3, Link(3, Link(4, Link(4, Link(3, Link(3))))))
    """
```

```
if s is Link.empty:
    return 0
elif s.rest is Link.empty:
    ____________ = ____________________________________________
    return ____________________________________________________
elif ____________________________________________________________:
    return double_up(_____________________________________________)
else:
    ____________ = ____________________________________________
    return ____________________________________________________
```

```
if s is Link.empty:
    return 0
elif s.rest is Link.empty:
    s.rest = Link(s.first)
    return 1
elif s.first == s.rest.first:
    return double_up(s.rest.rest)
else:
    s.rest = Link(s.first, s.rest)
    return 1 + double_up(s.rest.rest)
```