# Decomposition

## Announcements

## Today’s plan

- Midterm advice
- More Linked Lists
- Environment Diagram practice

## How to practice for the Midterm

- Spend 30 minutes each day over Spring Break looking over past topics
  - Do you understand them?
  - Skim through the lab / HW / project
- Friday of Spring break is when you should start doing practice Midterms
  - Start early and do one a day!
- Remember to look back on past midterms that you’ve done!
  - Learning is done when you look at your mistakes

## Recap: Doubling A List

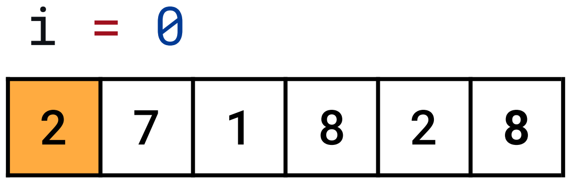
```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    ________________:
        if ____ == v:
            s.insert(i+1, v)
            ______
        else:
            i += 1
```

## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```

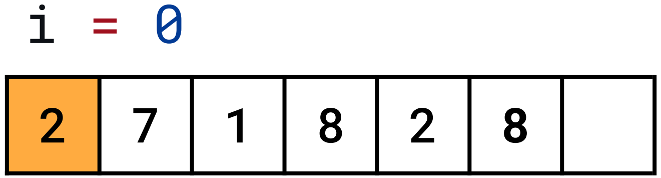
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



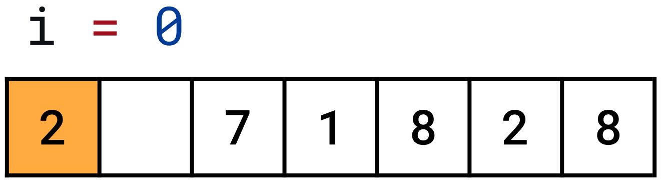
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



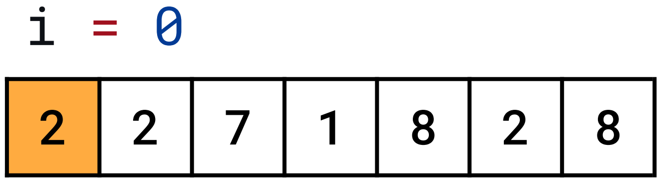
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



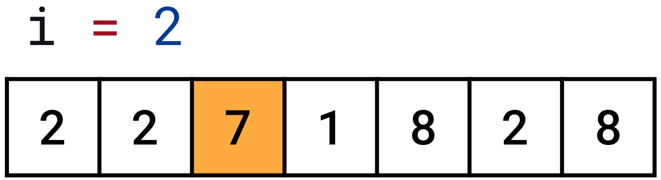
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



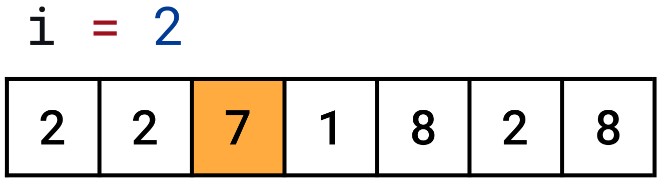
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



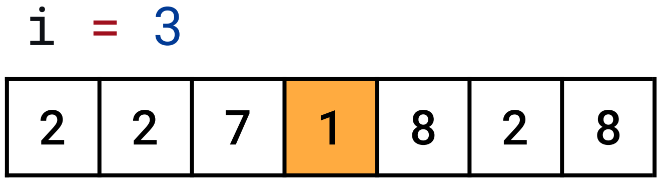
# Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



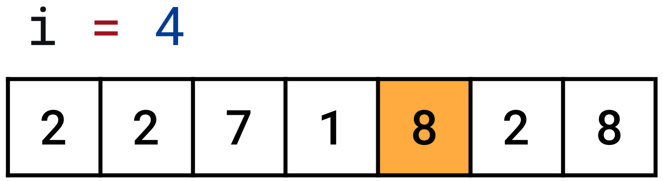
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



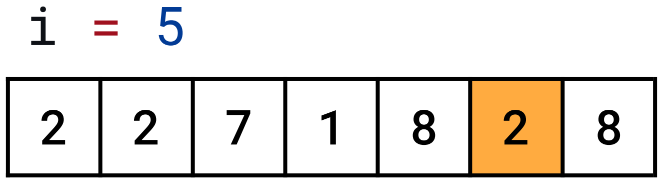
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



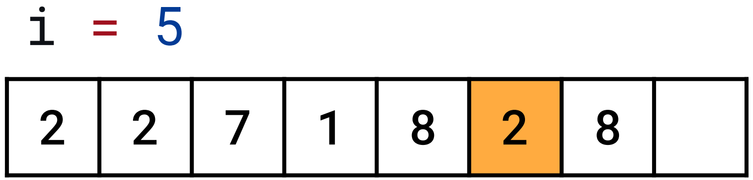
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



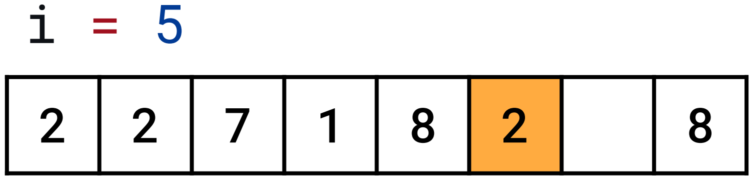
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



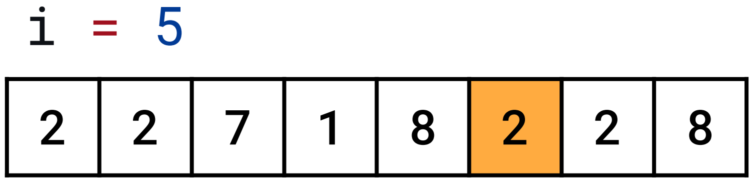
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



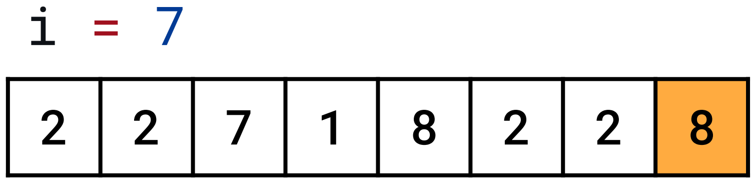
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



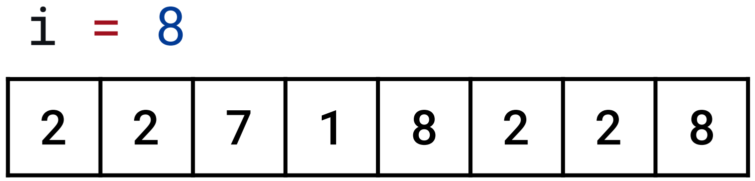
## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



## Recap: Doubling A List

```
def double(s, v):
    """Insert another v after each v in list s.
    >>> s = [2, 7, 1, 8, 2, 8]
    >>> double(s, 8)
    >>> s
    [2, 7, 1, 8, 8, 2, 8, 8]
    """
    i = 0
    while i < len(s):
        if s[i] == v:
            s.insert(i+1, v)
            i += 2
        else:
            i += 1
```



## Recap: Doubling A Linked List

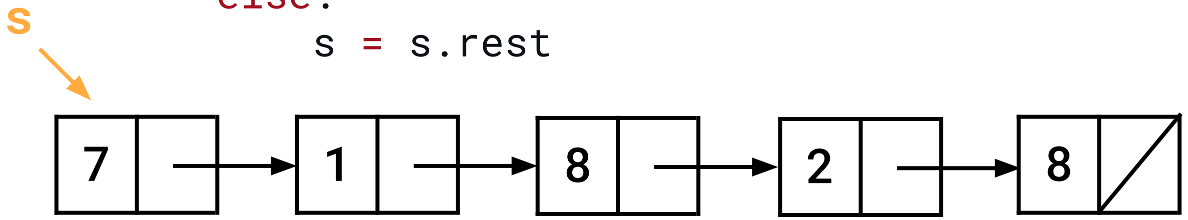
```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(7, Link(1, Link(8, Link(2, Link(8)))))
    >>> double_link(s, 8)
    >>> print(s)
    <7 1 8 8 2 8 8>
    """
    __________________________:
        if ________ == v:
            s.rest = Link(v, s.rest)
            ________________
        else:
            s = s.rest
```

## Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(7, Link(1, Link(8, Link(2, Link(8)))))
    >>> double_link(s, 8)
    >>> print(s)
    <7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```

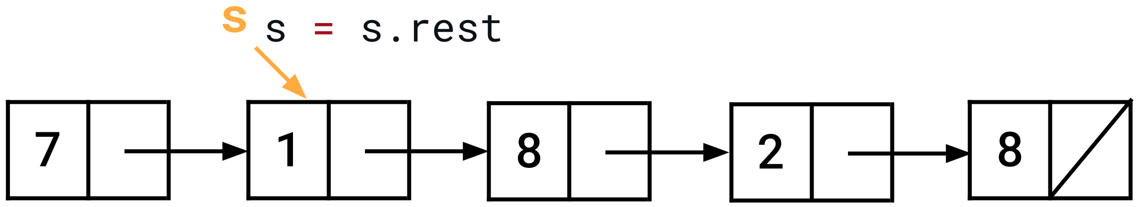
## Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



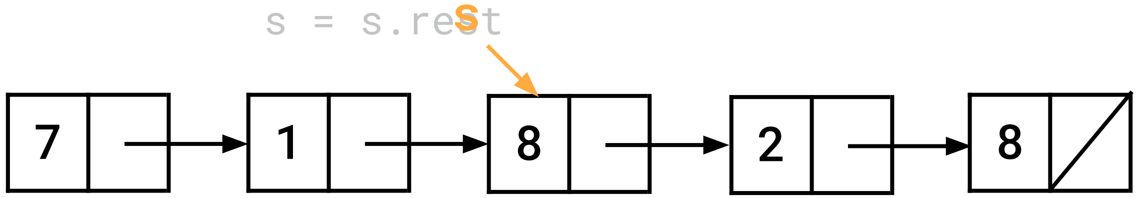
## Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



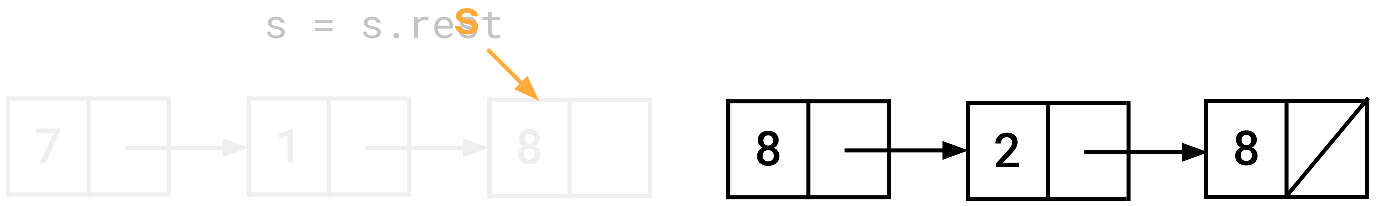
# Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



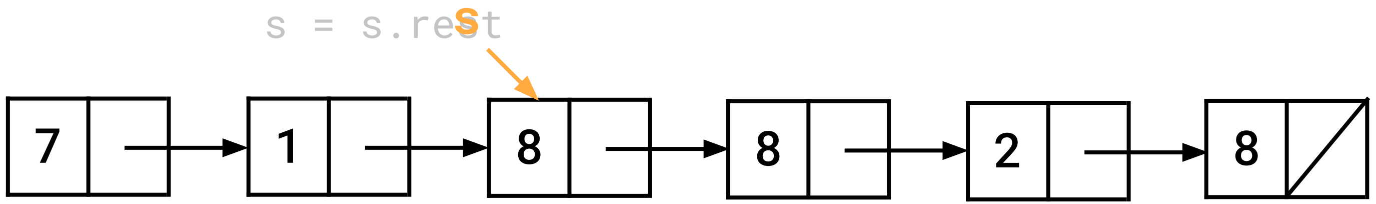
# Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



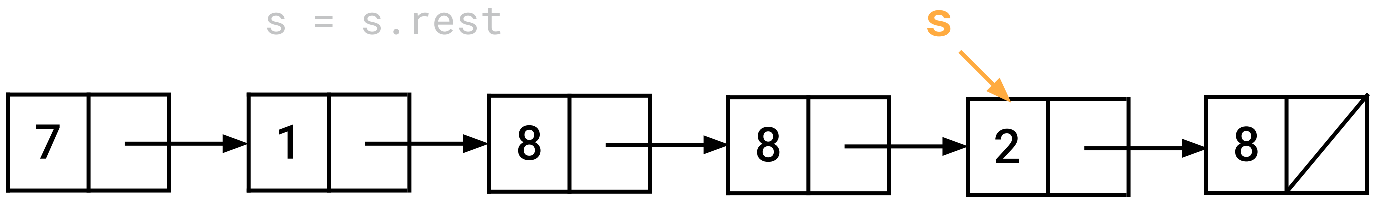
## Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



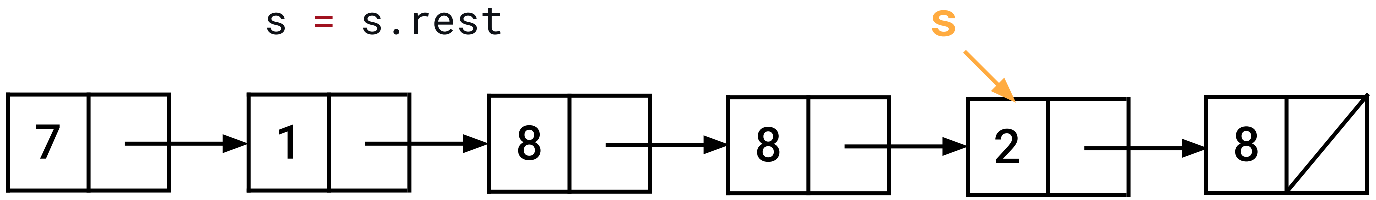
# Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



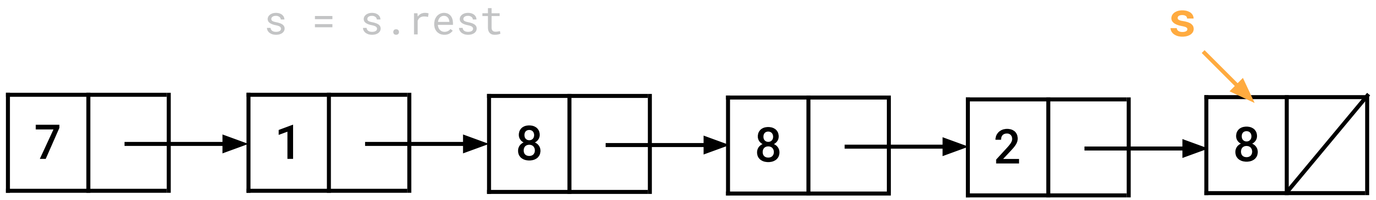
# Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



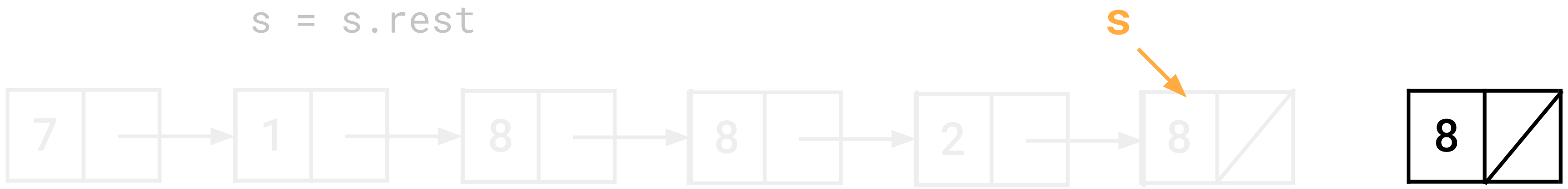
# Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



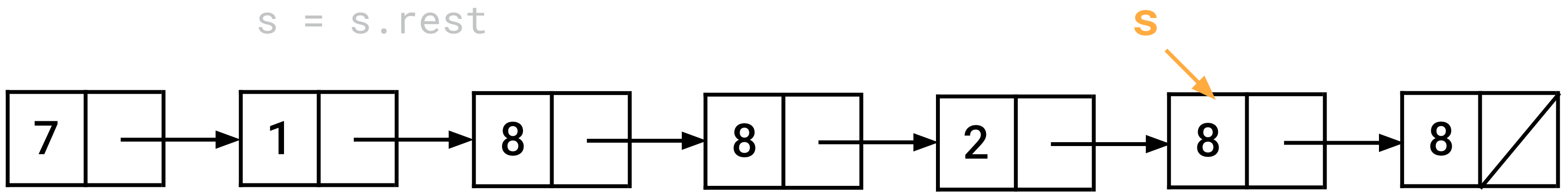
# Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



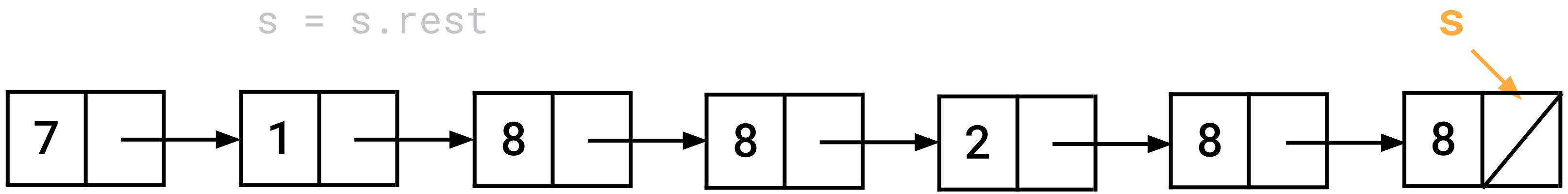
## Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```



## Recap: Doubling A Linked List

```
def double_link(s, v):
    """Insert another v after each v in linked list s.
    >>> s = Link(2, Link(7, Link(1, Link(8, Link(2, Link(8))))))
    >>> double_link(s, 8)
    >>> print(s)
    <2 7 1 8 8 2 8 8>
    """
    while s is not Link.empty:
        if s.first == v:
            s.rest = Link(v, s.rest)
            s = s.rest.rest
        else:
            s = s.rest
```

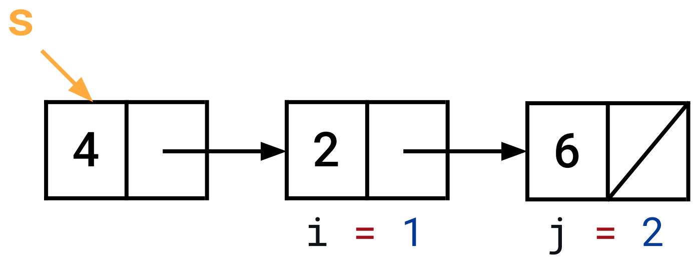


## Takeaways

- Normal list takes longer due to the `.insert` call
    - Has to shift all the items over
- Linked list is quicker
    - Just create a new link and shift pointers

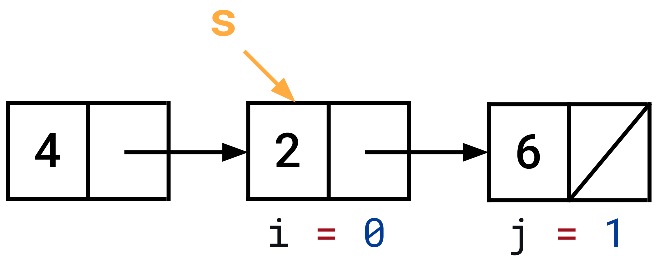
# Slicing a Linked List

```
def slice_link(s, i, j):
    """Return a new linked list containing elements from i:j.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> slice_link(evens, 1, 100)
    Link(2, Link(6))
    >>> slice_link(evens, 1, 2)
    Link(2)
    >>> slice_link(evens, 0, 2)
    Link(4, Link(2))
    >>> slice_link(evens, 1, 1) is Link.empty
    True
    """
    assert i >= 0 and j >= 0 and j >= i
    if _________ or s is Link.empty:
        return Link.empty
    elif ________:
        return Link(s.first, _________________)
    else:
        return slice_link(s.rest, ____ , ____ )
```



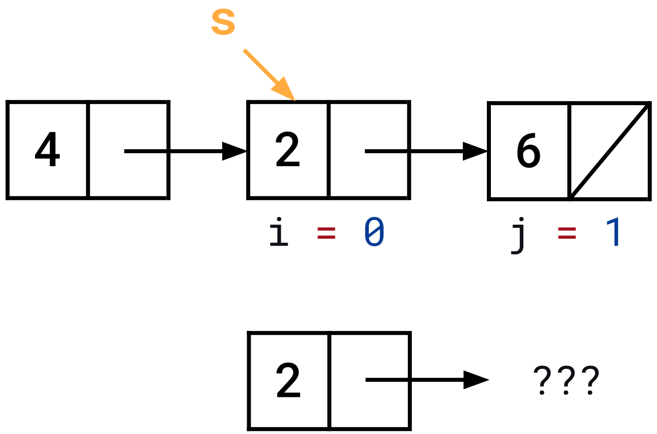
## Slicing a Linked List

```
def slice_link(s, i, j):
    """Return a new linked list containing elements from i:j.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> slice_link(evens, 1, 100)
    Link(2, Link(6))
    >>> slice_link(evens, 1, 2)
    Link(2)
    >>> slice_link(evens, 0, 2)
    Link(4, Link(2))
    >>> slice_link(evens, 1, 1) is Link.empty
    True
    """
    assert i >= 0 and j >= 0 and j >= i
    if _________ or s is Link.empty:
        return Link.empty
    elif ________:
        return Link(s.first, _________________)
    else:
        return slice_link(s.rest, i-1, j-1)
```



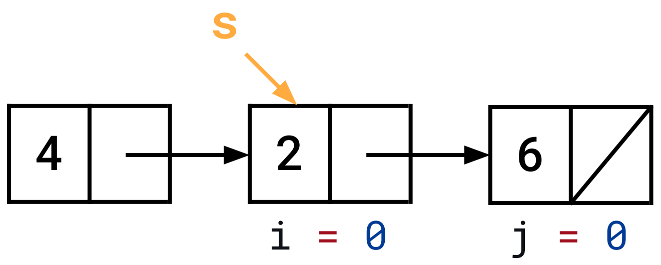
# Slicing a Linked List

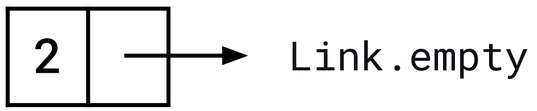
```
def slice_link(s, i, j):
    """Return a new linked list containing elements from i:j.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> slice_link(evens, 1, 100)
    Link(2, Link(6))
    >>> slice_link(evens, 1, 2)
    Link(2)
    >>> slice_link(evens, 0, 2)
    Link(4, Link(2))
    >>> slice_link(evens, 1, 1) is Link.empty
    True
    """
    assert i >= 0 and j >= 0 and j >= i
    if _________ or s is Link.empty:
        return Link.empty
    elif i == 0:
        return Link(s.first, slice_link(s.rest, i, j-1))
    else:
        return slice_link(s.rest, i-1, j-1)
```



# Slicing a Linked List

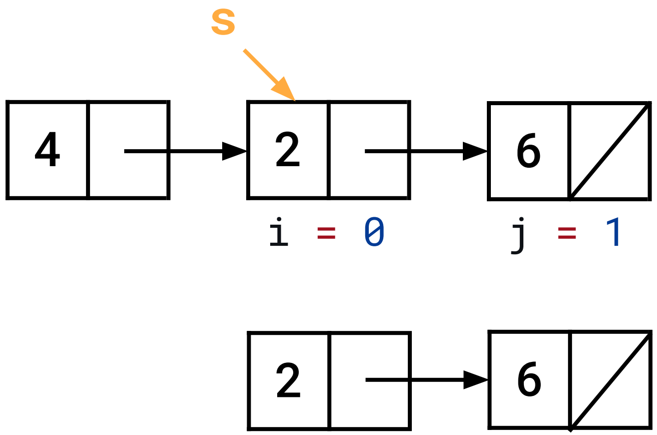
```
def slice_link(s, i, j):
    """Return a new linked list containing elements from i:j.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> slice_link(evens, 1, 100)
    Link(2, Link(6))
    >>> slice_link(evens, 1, 2)
    Link(2)
    >>> slice_link(evens, 0, 2)
    Link(4, Link(2))
    >>> slice_link(evens, 1, 1) is Link.empty
    True
    """
    assert i >= 0 and j >= 0 and j >= i
    if j == 0 or s is Link.empty:
        return Link.empty
    elif i == 0:
        return Link(s.first, slice_link(s.rest, i, j-1))
    else:
        return slice_link(s.rest, i-1, j-1)
```





# Slicing a Linked List

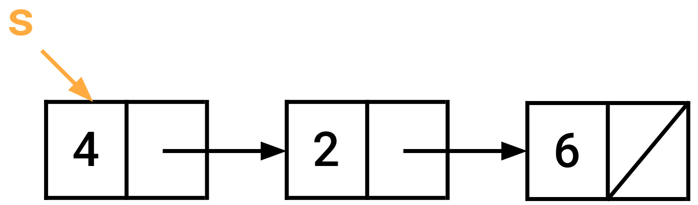
```
def slice_link(s, i, j):
    """Return a new linked list containing elements from i:j.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> slice_link(evens, 1, 100)
    Link(2, Link(6))
    >>> slice_link(evens, 1, 2)
    Link(2)
    >>> slice_link(evens, 0, 2)
    Link(4, Link(2))
    >>> slice_link(evens, 1, 1) is Link.empty
    True
    """
    assert i >= 0 and j >= 0 and j >= i
    if j == 0 or s is Link.empty:
        return Link.empty
    elif i == 0:
        return Link(s.first, slice_link(s.rest, i, j-1))
    else:
        return slice_link(s.rest, i-1, j-1)
```



# Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = ______________________
        s.first = _______________
        s.rest = second
    elif ____________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

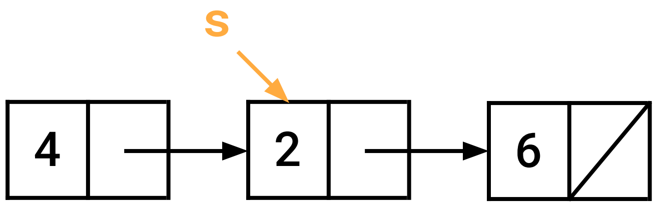
x = 8     i = 1



# Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = ______________________
        s.first = _______________
        s.rest = second
    elif ____________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

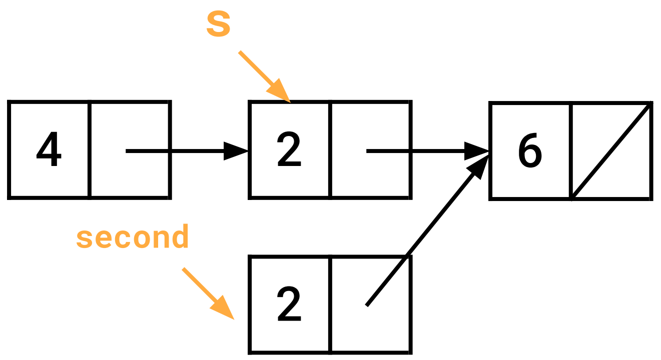
x = 8     i = 0



## Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif ___________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

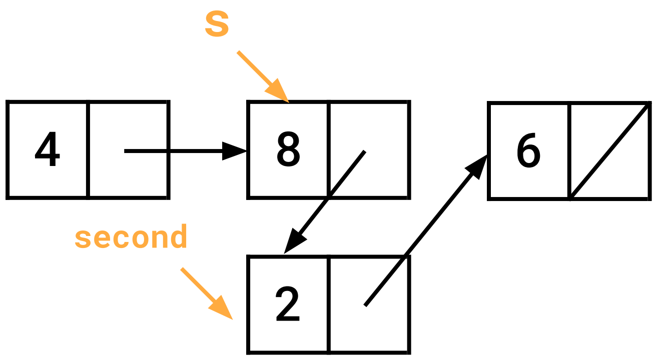
```
x = 8        i = 0
```



# Inserting into a Linked List

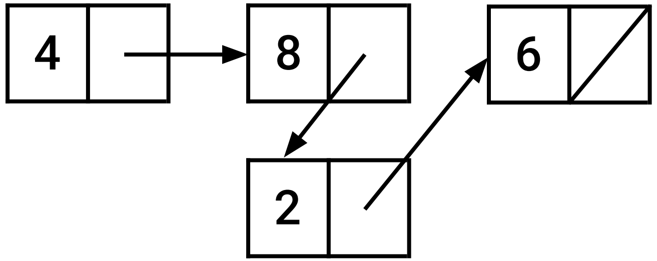
```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif _________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

```
x = 8        i = 0
```



# Inserting into a Linked List

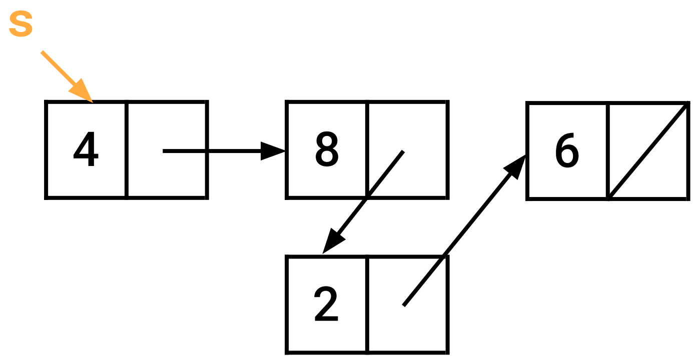
```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif __________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```



## Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif __________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

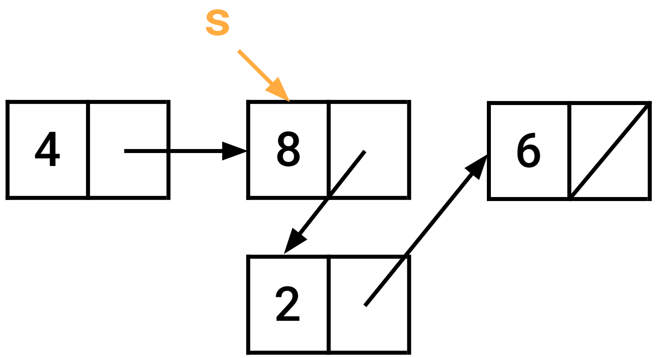
x = 10     i = 4



# Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif __________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

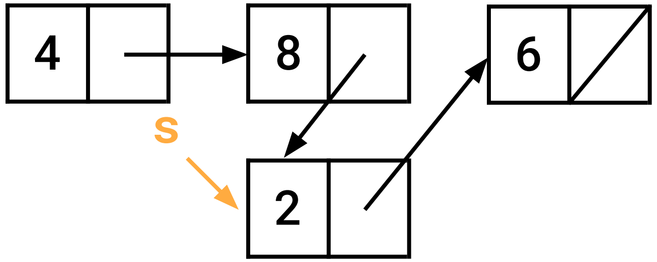
```
x = 10     i = 3
```



# Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif ___________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

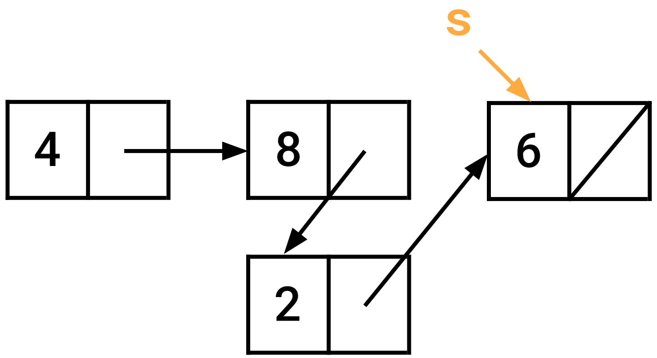
```
x = 10      i = 2
```



# Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif __________________________________:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

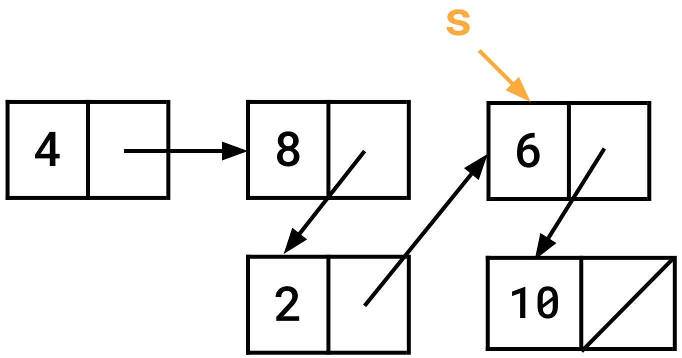
x = 10     i = 1



# Inserting into a Linked List

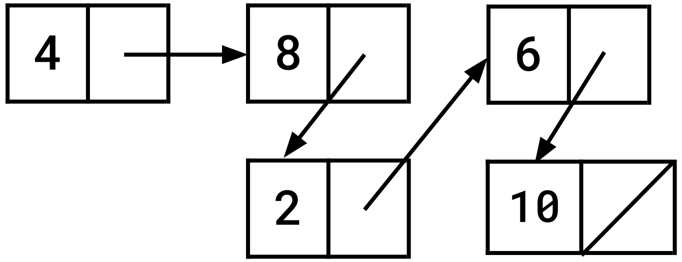
```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif i == 1 and s.rest is Link.empty:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

x = 10    i = 1



## Inserting into a Linked List

```
def insert_link(s, x, i):
    """Insert x into linked list s at index i.
    >>> evens = Link(4, Link(2, Link(6)))
    >>> insert_link(evens, 8, 1)
    >>> insert_link(evens, 10, 4)
    >>> insert_link(evens, 12, 0)
    >>> insert_link(evens, 14, 10)
    Index out of range
    >>> print(evens)
    <12 4 8 2 6 10>
    """
    if s is Link.empty:
        print('Index out of range')
    elif i == 0:
        second = Link(s.first, s.rest)
        s.first = x
        s.rest = second
    elif i == 1 and s.rest is Link.empty:
        s.rest = Link(x)
    else:
        insert_link(s.rest, x, i-1)
```

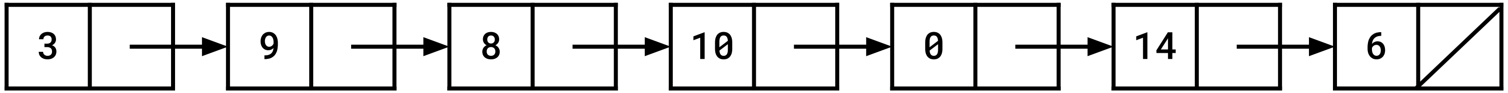


# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.

def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if ______________________________________:
            _____________________________________
    _____________________________________________
```

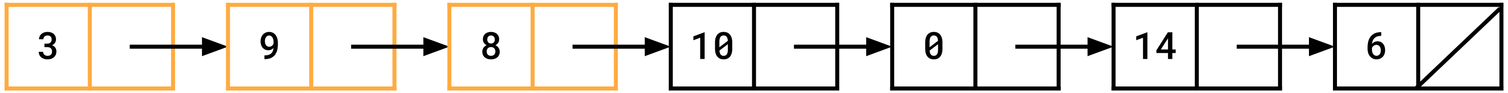


# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.

def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6)))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if _____________________________________:
            _____________________________________
    _____________________________________________
```

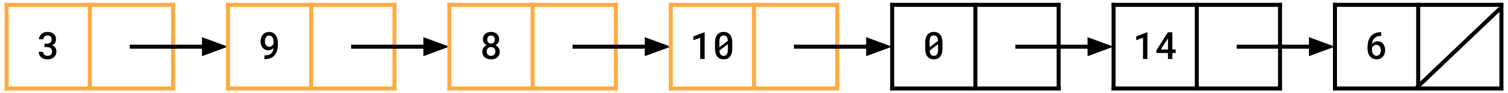


# Spring 2023 Midterm 2 Question 3(b)

Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link instance. It prints all of the prefix sums of s that are multiples of 10 in increasing order of the length of the prefix.

```
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if _____________________________________:
            _____________________________________
    _____________________________________________
```

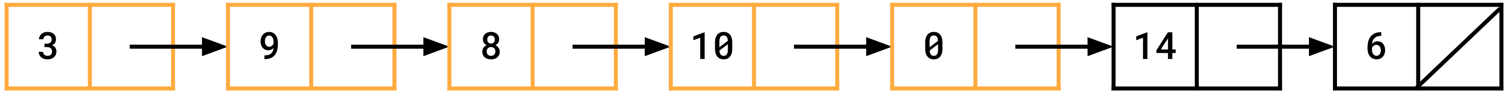


# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.

def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if ______________________________________:
            _____________________________________
    ______________________________________________
```

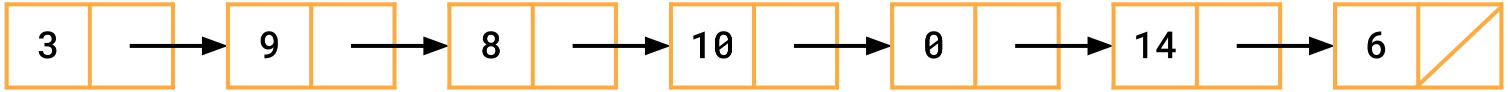


## Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.

def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6)))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if _____________________________________:
            _____________________________________
    _____________________________________________
```



## Spring 2023 Midterm 2 Question 3(b)

Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link instance. It prints all of the prefix sums of s that are multiples of 10 in increasing order of the length of the prefix.

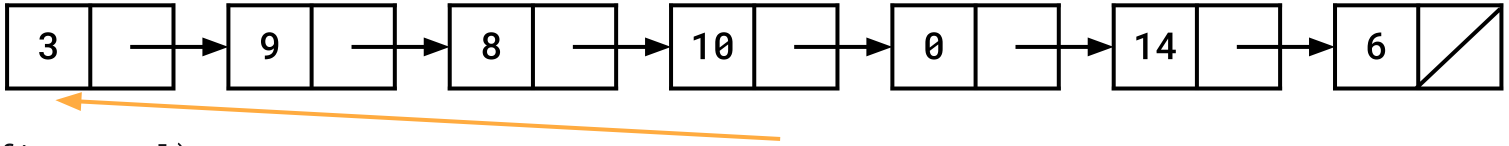
```python
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```

## Spring 2023 Midterm 2 Question 3(b)

Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link instance. It prints all of the prefix sums of s that are multiples of 10 in increasing order of the length of the prefix.

```
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```



suffix

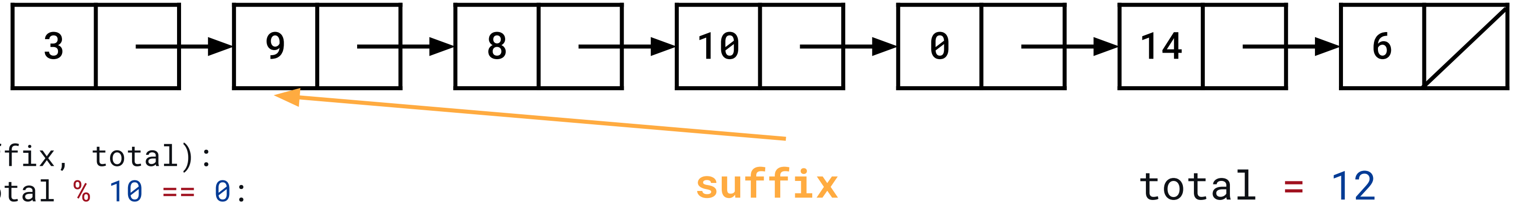
total = 3

# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.
```

```python
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```



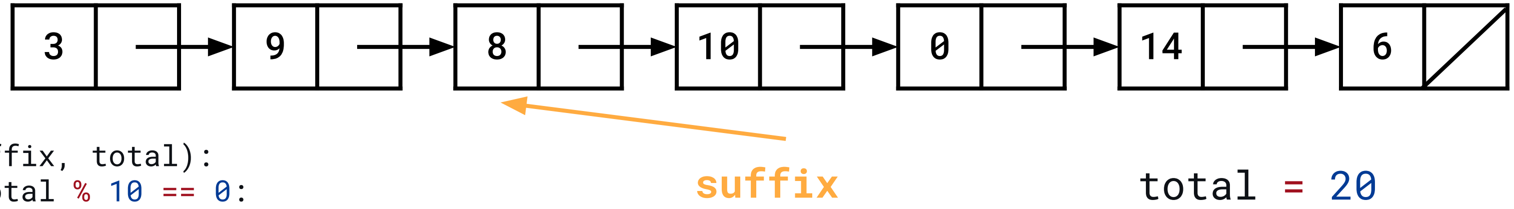
total = 12

# Spring 2023 Midterm 2 Question 3(b)

Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link instance. It prints all of the prefix sums of s that are multiples of 10 in increasing order of the length of the prefix.

```python
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```

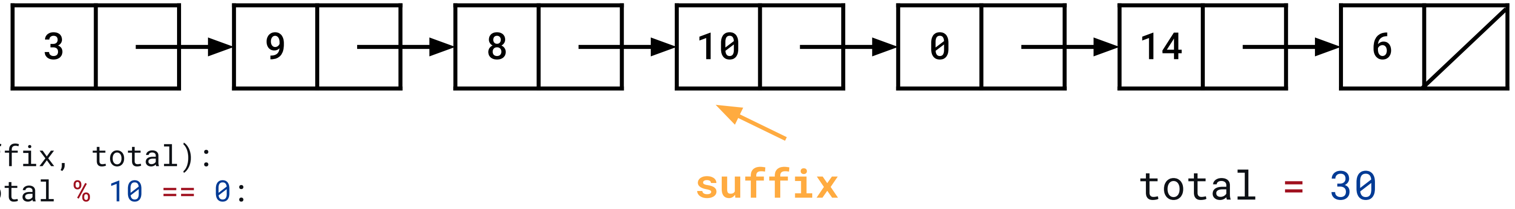


# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.
```

```
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```



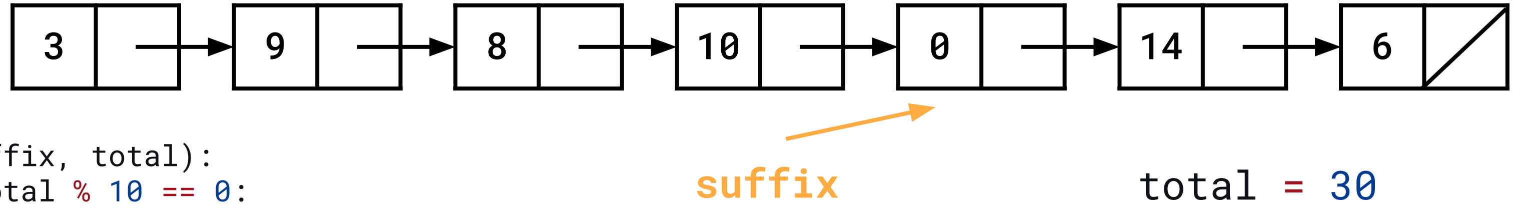
total = 30

# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.
```

```
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```

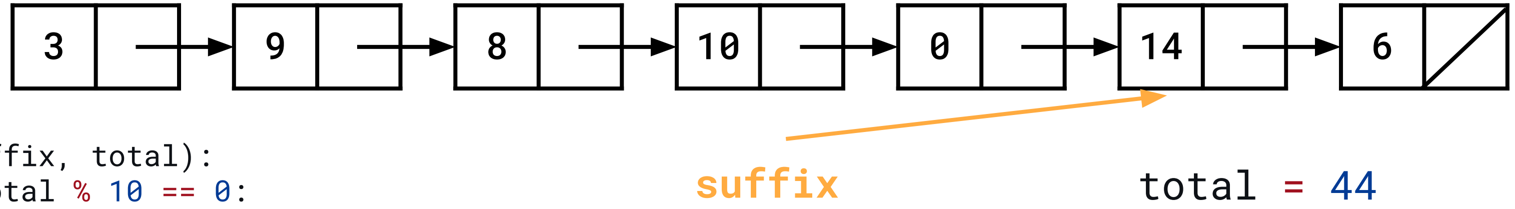


# Spring 2023 Midterm 2 Question 3(b)

Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link instance. It prints all of the prefix sums of s that are multiples of 10 in increasing order of the length of the prefix.

```
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6))))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```



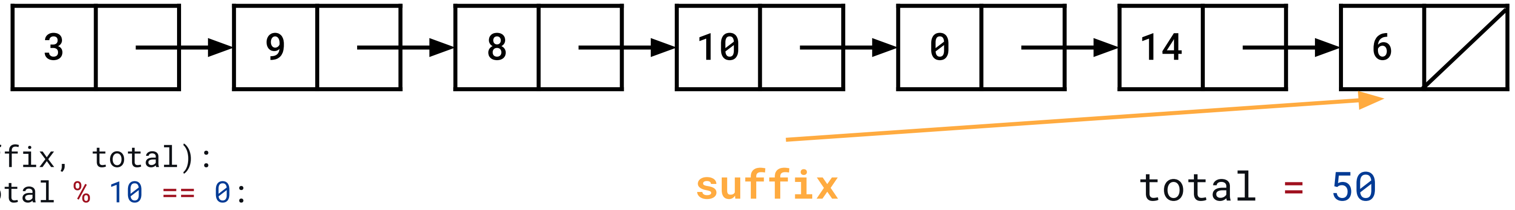
total = 44

# Spring 2023 Midterm 2 Question 3(b)

```
Definition. A prefix sum of a sequence of numbers is the sum of the first n elements for
some positive length n.

Implement tens, which takes a non-empty linked list of numbers s represented as a Link
instance. It prints all of the prefix sums of s that are multiples of 10 in increasing
order of the length of the prefix.
```

```python
def tens(s):
    """Print all prefix sums of Link s that are multiples of ten.
    >>> tens(Link(3, Link(9, Link(8, Link(10, Link(0, Link(14, Link(6)))))))
    20
    30
    30
    50
    """
    def f(suffix, total):
        if total % 10 == 0:
            print(total)
        if suffix is not Link.empty:
            f(suffix.rest, total + suffix.first)
    f(s.rest, s.first)
```



total = 50

## Surf’s Up (Su25 Midterm Q2)

Answer the questions about the code below. Use the free space or scratch paper to draw the diagram to help you answer the questions, however any drawn diagrams will not be graded.

```python
def wave(twin):
    n = 3
    def twin():
        print(wave, n)
        return print('Shred') or n
        print('Wipeout')
    return twin()

def twin():
    def twin(wave):
        while wave(twin):
            return print('Riptide')
        print('Tide')
    return twin

twin = twin()(lambda twin: print('Twin Wave') or wave(twin))
```