1. You Complete Me (Sp15 Midterm 1 3a)

(a) (4 pt) Implement the longest_increasing_suffix function, which returns the longest suffix (end) of a positive integer that consists of strictly increasing digits.

```python
def longest_increasing_suffix(n):
    """Return the longest increasing suffix of a positive integer n."

    >>> longest_increasing_suffix(63134)
    134
    >>> longest_increasing_suffix(233)
    3
    >>> longest_increasing_suffix(5689)
    689
    >>> longest_increasing_suffix(568901) # 01 is the suffix, displayed as 1
    1
    """

    n, suffix, k = 10, 0, 1

    while n:
        n, last = n // 10, n % 10

        if last < m:
            m, suffix, k = last, suffix + k * last, 10 * k
        else:
            return suffix

    return suffix
```

2. A Highly Intelligent Animal (Su15 Midterm 1 Q4c)

A number \( n \) contains a sandwich if a digit in \( n \) is surrounded by two identical digits. For example, the number 242 contains a sandwich because 4 is surrounded by 2 on both sides. 1242 also contains a sandwich, while 12532 does not contain a sandwich.

Implement the `sandwich(n)` function, which takes in a nonnegative integer \( n \). It returns True if \( n \) contains a sandwich and False otherwise. If \( n \) has fewer than three digits, it cannot contain a sandwich.

```python
def sandwich(n):
    """Returns True if n contains a sandwich and False otherwise."""

    >>> sandwich(416263)  # 626
    True
    >>> sandwich(5050)    # 505 or 050
    True
    >>> sandwich(4441)    # 444
    True
    >>> sandwich(1231)    # False
    False
    >>> sandwich(55)      # False
    False
    >>> sandwich(4456)    # False
    False

    tens, ones = (n // 10) & 10, n % 10

    n = n // 100

    while n > 0:
        if n % 10 == ones:
            return True

        else:
            tens, ones = n % 10, tens

            n = n // 10

    return False
```
3. Digit Fidget (Fa15 Midterm 1 Q3c)

(3 pt) Implement luhn_sum. The Luhn sum of a non-negative integer \( n \) adds the sum of each digit in an even position to the sum of doubling each digit in an odd position. If doubling an odd digit results in a two-digit number, those two digits are summed to form a single digit. **You may not use recursive calls or call find_digit in your solution.**

```python
def luhn_sum(n):
    """Return the Luhn sum of n."

    >>> luhn_sum(135)  # 1 + 6 + 5
    12
    >>> luhn_sum(185)  # 1 + (1+6) + 5
    13
    >>> luhn_sum(138743)  # From lecture: 2 + 3 + (1+6) + 7 + 8 + 3
    30
    """

def luhn_digit(digit):

    x = digit * multiplier

    return (x // 10) + x % 10

total, multiplier = 0, 1

while n:

    n, last = n // 10, n % 10

    total = total + luhn_digit(last)

    multiplier = 3 - multiplier

return total
```
4. Dog Goes Woof (Fa13 Midterm 1 Q1)

1. (12 points) Dog Goes Woof

For each of the following call expressions, write the value to which it evaluates and what would be output by the interactive Python interpreter. The first two rows have been provided as examples.

Assume that you have started Python 3 and executed the following statements:

```python
from operator import add, mul

def square(x):
    return mul(x, x)

def dog(bird):
    def cow(tweet, moo):
        woof = bird(tweet)
        print(moo)
        return woof
    return cow

cat = dog(square)
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Evaluates to</th>
<th>Interactive Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>square(5)</code></td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td><code>1/0</code></td>
<td>Error</td>
<td>Error</td>
</tr>
<tr>
<td><code>add(square(2), mul(3, 4))</code></td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>
| `print(print(print(2)))`    | None         | 2
|                             |              | None None          |
| `cat(3, 4)`                 | 9            | 4
|                             |              | 9                  |
| `square(cat(5))`            | Error        | Error              |
| `cat(square(2), print(5))`  | 16           | 5
|                             |              | None None 16       |
| `cat(print(square(3)), 8)`  | Error        | 9
|                             |              | Error              |
5. Supernatural (Sp15 Midterm 1 Q2a)

2. (14 points) Supernatural

(a) (6 pt) Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You may not need to use all of the spaces or frames.

A complete answer will:
- Add all missing names and parent annotations to all local frames.
- Add all missing values created or referenced during execution.
- Show the return value for each local frame.

**Remember:** Do not add a new frame when calling a built-in function (such as abs).

```
batman, superman, Ivy = 1, -2, -3

def namana(batman):
    while batman(superman) > Ivy:
        def batman(joker):
            return Ivy
    return -Ivy

def joker(superman):
    if superman(batman):
        Ivy = -batman
    return namana

joker(abs)(abs)
```

Global frame:
- batman: 1
- superman: -2
- Ivy: -3

Local frames:
1. `f1: joker [parent=Global]`
   - superman: 
   - Ivy: -1
2. `f2: namana [parent=Global]`
   - batman: 
3. `f3: batman [parent=f2]`
   - joker: -2

Return Value:
- 3
6. Envy, Iron, Mint (Fa14 Midterm 1 Q2a)

2. (14 points) Envy, Iron, Mint

(a) (6 pt) Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You may not need to use all of the spaces or frames.

A complete answer will:

- Add all missing names, labels, and parent annotations to all local frames.
- Add all missing values created during execution.
- Show the return value for each local frame.

```python
def peace(today):
    harmony = love + 2
    return harmony + today(love=1)

def joy(peace):
    peace, love = peace+2, peace+1
    return love // harmony

love, harmony = 3, 2
peace = joy
```

Global frame

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>peace</td>
<td></td>
</tr>
<tr>
<td>joy</td>
<td></td>
</tr>
<tr>
<td>love</td>
<td>3</td>
</tr>
<tr>
<td>harmony</td>
<td>2</td>
</tr>
</tbody>
</table>

func peace(today) [parent=Global]

func joy(peace) [parent=Global]

f1: peace [parent=Global]

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>today</td>
<td></td>
</tr>
<tr>
<td>harmony</td>
<td>5</td>
</tr>
<tr>
<td>Return Value</td>
<td>7</td>
</tr>
</tbody>
</table>

f2: joy [parent=Global]

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>peace</td>
<td>6</td>
</tr>
<tr>
<td>love</td>
<td>5</td>
</tr>
<tr>
<td>Return Value</td>
<td>2</td>
</tr>
</tbody>
</table>

f3: ______ [parent=______]

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Value</td>
<td></td>
</tr>
</tbody>
</table>