Recursion Continued

COSC 101: Intro to Computing I
November 27, 2017
Mini Review
What is recursion?
Why use recursion?
What is a base case?
How does the function reach the base case?
What is infinite recursion?
How to write a recursive function?

1. Write the docstring first (*it really does help*)

2. Figure out the base case (often 0, 1, or "")

3. For the recursive case:
   i. **Divide**: Break into two pieces:
      a. *Simple* to handle now
      b. *Harder* a smaller version of the same problem
   ii. **Recurse**: Make a recursive call with the harder piece and “*have faith*” that it will come together correctly
   iii. **Combine**: Put the result of the recursive call and the simple piece together into a complete solution
New Exercises
What is the output of the following code?

```python
def mystery(a):
    if len(a) <= 2:
        return a
    x = len(a)//2
    return mystery(a[:x] + a[x+1:]) + a[x]

print(mystery('1234567'))
```

1726354
What is the output of the following code?

```python
def mystery2(a):
    if len(a) <= 2:
        return a
    x = len(a)//2
    return mystery2(a[:x]) + a[x] + mystery2(a[x+1:])

print(mystery2('1234567'))
```

1234567
What is the output of the following code?

```python
def mystery3(a):
    if len(a) <= 2:
        return a
    x = len(a)//2
    return a[x] + mystery3(a[x+1:])
    + mystery3(a[:x])

print(mystery3('1234567'))
```

4675231
Write a function `no_duplicate_e` that takes a string `s` and returns the string with all 'ee' replaced with 'e'.

```python
def no_duplicate_e(s):
    ''' (str) -> str
    Returns the string s with any 'ee' s replaced with 'e's.
    >>>no_duplicate_e('heello')
    'hello'
    >>>no_duplicate_e('He said, "Wheeeeee!".')
    'He said, "Whe!"."
    >>>no_duplicate_e('')
    ''
    ...
```
Write a function `no_duplicate_e` that takes a string `s` and returns the string with all 'ee' replaced with 'e'.

def no_duplicate_e(s):
    if s == '':
        return s

    simple = s[0:2]  # divide
    harder = s[1:]

    result = no_duplicate_e(harder)  # recurse

    if simple == 'ee':
        return result  # combine

    return simple[0] + result
Write a function `no_duplicate_e` that takes a string `s` and returns the string with all `'ee'` replaced with `'e'`.

```python
def no_duplicate_e_shorter(s):
    if s == '':
        return s
    if s[0:2] == 'ee':
        return no_duplicate_e_shorter(s[1:])
    return s[0] + no_duplicate_e_shorter(s[1:])
```
Write a function `mirror` that takes a string `s` and returns the string “mirrored”. For example 'ah' becomes 'ahha'.

def mirror(s):
    ''' (str) -> str
    Returns the a mirrored version of s.
    >>>mirror('madeline')
    'madelineeniledam'
    >>>mirror('computer science!')
    'computer science!!eceics retupmoc'
    >>>mirror('"
    ''
    '''
    ...
Write a function `mirror` that takes a string `s` and returns the string “mirrored”. For example 'ah' becomes 'ahha'.

```python
def mirror(s):
    if s == '':
        return s
    return s[0] + mirror(s[1:]) + s[0]
```
Write a function `duplicate` that takes a string `s` and returns a version of `s` where every letter is duplicated. For example, `'ah'` becomes `'aahh'`.

```python
def duplicate(s):
    ''' (str) -> str
    Returns the version of s where every letter is duplicated.
    >>> duplicate('madeline')
    'mmaaddeelliinnee'
    >>> duplicate('whattt??!!')
    'wwhhaattttttt????!!!!!'
    >>> mirror('')
    ''
    '''
```

Write a function `duplicate` that takes a string `s` and returns a version of `s` where every letter is duplicated. For example, `'ah'` becomes `'aahh'`.

```python
def duplicate(s):
    if s == '':
        return s
    return s[0]*2 + duplicate(s[1:])
```
Write a function `is_pal` that takes a string `s` and returns `True` if `s` is a palindrome and `False` otherwise.

```python
def is_pal(s):
    ''' (str) -> str
    Returns True is s is a palindrome, and False if s is not a palindrome.
    >>>is_pal('madeline')
    False
    >>>is_pal('kayak')
    True
    >>>is_pal('')
    True
    ...
```
Write a function `is_pal` that takes a string `s` and returns `True` if `s` is a palindrome and `False` otherwise.

```python
def is_pal(s):
    if s == '':
        return True
    return s[0] == s[-1] and is_pal(s[1:])
```