CS 1110 Review: Final

This worksheet contains various problems for you to practice. We will go over only some problems during the final review session and will let you take this worksheet home to practice more.

Call Frames

Draw the entire call stack for skip('abc')

```
def skip(s):
    """Returns: copy of s odd (from end) skipped"""
1   result = ""
2   if (len(s) % 2 == 1):
3       result = skip(s[1:])
4   elif len(s) > 0:
5       result = s[0]+skip(s[1:])
6   return result
```

Draw the call stack for sumStringList([‘ 8’, ’5’, ‘ 70 ’])

```
def sumStringList(li):
"""Returns: the sum of a list of strings.
Precondition: li is list of strings of digits with
possible white space before or after the digits.
ex: li = [‘ 8 ’, ‘32’, ‘ 1’] returns 41"""
counter = 0
for x in li:
        x = x.strip()
        counter+=int(x)
return counter
```

Draw the diagrams for the 2 object folders and the class folder when you execute:

```
class Cornellian(object):
    """Instance attributes:
    _cuid: Cornell id [int > 0]
    _name: full name [nonempty str]"""

    NEXT = 1 # Class Attribute

    def _assignCUID(self):
        """Assigns _cuid to next Cornell id"""
        self._cuid = Cornellian.NEXT
        Cornellian.NEXT = Cornellian.NEXT+1

    def __init__(self, n):
        """Initializer: Cornellian with name n."""
        self._name = n
        self._assignCUID()
        a = Cornellian("Alice")
        b = Cornellian("Bob")


>>>a = Cornellian("Alice")
>>>b = Cornellian("Bob")
```

Diagram the class folder and the constructor call frames for the following two classes when you execute the following code:

```
class X():
    b = 1110
    a = 5
    def __init__(self, d):
        self.a  = self.b
        self.c = d
class Y(X):
    b = 10
    def __init__(self, d, a):
        super().__init__(a)
        self.b=d

>>>y = X(5)
>>>x = Y(1110,5)
```

Classes

```
class Book():
    """Instance is a book that is currently being read
    Attributes:
        title [str]: title of the book
        sequel [Book]: sequel to the book, None if nonexistent
    pages_left [int]: number of pages still unread"""

    def __init__(self,                                            ):
        """Initializer for class Book
        Default value for sequel is None
        Default value for pages_left is 0 """








    def __eq__(self, b):
        """Returns: a boolean based on whether Book b has the same
        attributes as this instance"""






    def __str__(self):
        """Format: title (sequel, pages_left) """
        sequel_str = ""
        if self.sequel != None:
        sequel_str = str(self.sequel)
        return (self.title + "␣(" + sequel_str + ",␣" +
        str(self.pages_left) + ")")

    def readPages(self, n):
        """Read n number of pages in the book until the end"""
```

What are the outputs when running the following pieces of code?

```
>>>b1 = Book("Eldest")
>>>b2 = Book("Eragon", b1, 30)
>>>b3 = Book("Eldest")
>>>b4 = b1
>>>
>>>print(b1 == b2)
>>>print(b1 is b2)
>>>print(b1 == b3)
>>>print(b1 is b3)
>>>print(b1 == b4)
>>>print(b1 is b4)
>>>print()
>>>print(b1.sequel)
>>>print(b1.pages_left)
>>>print(b2.sequel)
>>>print(b2.pages_left)
>>>print(b2)
```

```
#class Complex with the following specification
class Complex():
    """Instance is a complex number, with real and imaginary parts
    Attributes:
    real: real portion of the number   [float]
    imag: imaginary portion of number  [float]"""

    # initializer
    def __init__(self, real, imag):
        """Initializes attributes with floats
        Precondition: real, imag can be ints or floats"""





    # implement +
    def __add__(self, other):




    # implement *
    def __mul__(self, other):
        """Note: (a + bi) * (c + di) = (ac - bd) + (ad + bc)i"""






    # implement str: Complex(1, -2) looks like "1.0 + -2.0i"
    def __str__(self):




    # implement ==
    def __eq__(self, other):




# subclass Real, instance is a real number
class Real(Complex):
    """Instance is a real number"""

    # initializer for Real
    def __init__(self, num):
        """Init for Real calls Complex __init__ method"""
```

What is printed when the following code is executed?

```
>>>a = Complex(1, 2)
>>>print (a.getReal())
>>>print (a.getImag())
>>>print (a)
>>>b = Complex(-1, -2)
>>>print (b.getReal())
>>>print (b.getImag())
>>>print (b)
>>>c = Real(3)
>>>print (c.getReal())
>>>print (c.getImag())
>>>print (c)
>>>
>>>print ("Operation␣results:␣")
>>>print (a + b)
>>>print (a - b)
>>>print (a * b)
>>>print (a == b)
>>>print (a == Complex(1, 2))
```