fullName:____________________________andrewID:____________________________ recitationLetter:_____

15-112 S23

Quiz9 version B (30 min)

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating. (You may discuss it only once the quiz has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper quiz, and we will not grade it.
- Please try to limit questions so as to not distract your peers. **We will answer two questions at most per person.** If you are unsure how to interpret a problem, take your best guess.
- Unless otherwise stated, you may not use any concepts (including builtin functions) which we have not covered in the notes in weeks 1-9. You may not use recursion.
- Assume almostEqual(x, y) and rounded(n) are both supplied for you. You must write all other helper functions you wish to use, unless we specify otherwise.

True/False [3pts ea, 9 total]

Consider the following code for TF1-TF3:

```
def Dog:
    pass
dog = Dog()
```

Mark each option as either True or False. (Fill in one circle for each question.)

TF1. `type(dog) == Dog`

○ True

○ False

TF2. `dog.hi = 'hi'` will crash

○ True

○ False

TF3. `dog` is a mutable object

○ True

○ False

CT1: Code Tracing [12pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```python
class A:
    x = 0
    def __init__(self, x, y):
        self.x = x
        self.y = y
        A.x += x + y

    def __repr__(self):
        return f'A({self.x},{self.y})'

    def __eq__(self, other):
        return isinstance(other, A) and (self.x == other.x) and (self.y == other.y)

    def f(self):
        self.y += self.x
        A.x += self.x

    def g(self):
        return A(self.x, self.y)

def ct1(x, y):
    a1 = A(x, y)
    a2 = a1
    a2.y = 10
    a3 = A(a1.x, 10)
    print(a1 == a3)
    print(a3.f())
    print(a3.g())
    print([a1, a2, a3, A.x])
ct1(4, 6)
```

Free Response 1 Part A: Chapter Class [35 points]

Write the Chapter class so that the following test code passes. Make sure to pay attention to the test cases to know whether each test is calling a method or attribute/property! Do not hardcode the test cases, but you may assume that the parameters are always legal.
Hint: You do not need class attributes, inheritance, or hash methods for this class!

```
def testChapterClass():
    chapterA = Chapter('I love CS', 30) # chapter title, # of pages
    assert(chapterA.title == 'I love CS')
    assert(chapterA.pages == 30)
    assert(str(chapterA) == "I love CS: 30 pages")
    chapterB = Chapter('So do I', 15)
    assert(chapterB.title == 'So do I')
    assert(chapterB.pages == 15)
    assert(str(chapterB) == "So do I: 15 pages")
    #Convert chapters in lists properly!
    assert(str([chapterA, chapterB]) ==
                        '[I love CS: 30 pages, So do I: 15 pages]')

    #A chapter is only equal to another chapter with the same title and # pages.
    assert(chapterA == Chapter('I love CS', 30))
    assert(chapterA != Chapter('I love CS', 30000))
    assert(chapterA != chapterB)
    assert(chapterA != 'I love CS') #Do not crash here!
    print('Chapter class passed!')
testChapterClass()
```

Begin your FR1 Part A Chapter class here or on the next page

You may begin or continue your Chapter class here

Free Response 1 Part B: Book Class [44 points]

Write the Book class so that the following test code passes. Note that your Book class will need to use your Chapter class (including some of its methods) in certain places. You may use your Chapter class from FR1A. (You do not need to rewrite it here.) Make sure to pay attention to the test cases to know whether each test is calling a method or attribute/property! Do not hardcode the test cases, but you may assume that the parameters are always legal (so, for example, chapter indexes are always in bounds).
Hint: You do not need class attributes, inheritance, or hash methods for this class!

```python
def testBookClass():
    chapterA = Chapter('I love CS', 30) # chapter title, # of pages
    chapterB = Chapter('So do I', 15)
    book1 = Book('CS is Fun', [chapterA, chapterB]) # book title, chapters
    assert(not isinstance(book1, Chapter)) #Books do NOT inherit from Chapters!
    assert(book1.getChapterCount() == 2)
    assert(book1.getPageCount() == 45)
    assert(book1.getChapter(0) == Chapter('I love CS', 30))
    assert(book1.getChapter(1) == Chapter('So do I', 15))
    assert(str(book1) == 'CS is Fun [I love CS: 30 pages, So do I: 15 pages]')

    #Let's make another book
    book2 = Book('The Short Book', [ Chapter('Quick Read', 5) ])
    assert(book2.getChapterCount() == 1)
    assert(book2.getPageCount() == 5)
    assert(book2.getChapter(0) == Chapter('Quick Read', 5))

    #A book is only equal to another book with the same title and chapter list
    assert(book1 != book2)
    assert(book1 != 42) #Do not crash here!
    book3 = Book('CS is Fun', [chapterA, chapterB])
    assert(book1 == book3)

    #We can add chapters to the end of a book
    book3.addChapter('Epilogue', 10) #NOTE: This does not take in a Chapter object
    assert(book1 != book3)
    assert(str(book3) ==
       'CS is Fun [I love CS: 30 pages, So do I: 15 pages, Epilogue: 10 pages]')
    assert(book3.getChapterCount() == 3)
    assert(book3.getPageCount() == 55)
    print('Book class passed!')
testBookClass()
```

Begin your FR1 Part B Book class answer here

You may begin or continue your Book class here

bonusCT: Code Tracing [2pts bonus]

This question is optional. Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
class A(object):
    def __init__(self, x):
        self.x = x
    def B(self, x):
        self.x += self.B(x)
        return x + self.x
    A = B
    def B(self, x):
        self.x += 2*x
        return x + self.x

class B(A):
    def A(self, x):
        return super().A(x) + super().B(x)

def f(a):
    return (a.A(3), a.x)

print(f(A(4)))
print(f(B(4)))
```