

Name: _____ Section: ____ Andrew Id: _____ a

15-112 Spring 2020 Quiz 4

Up to 25 minutes. No calculators, no notes, no books, no other paper, no computers.

No recursion

You may call `almostEqual(x, y)` and `roundHalfUp(d)` without writing them. Write everything else!

1. Code Tracing [25pts]

Indicate what the following code prints. Place your answer (and nothing else) in the box to the right.

```
def ct(a):
    b = a
    c = copy.copy(a)
    b[0] = 64
    a.append("wow")
    c[3] = 2 * c[3]
    b = b[:2] + ["hi"] + b[2:]
    c.pop(1)
    a[-2] = 112
    print("a:", a)
    print("b:", b)
    print("c:", c)

z = ["feb", 14, "2020", 180]
ct(z)
print("z:", z)
```

2. Free Response: Bookshelf class [45pts]

Write the Bookshelf class so the following test function passes (and do not hardcode anything, so any reasonably similar uses of the Bookshelf class would also work properly). We can add books to a shelf, or we can move them from one shelf to another.

Note: You may assume we will not try to move a book if it is not already on a shelf, BUT we will have to make sure it is moving to a shelf with enough space.

```
def testBookshelfClass():
    s1 = Bookshelf(5)
    assert(s1.books == []) #No books on the shelf
    assert(s1.getSpace() == 5) #There is space for 5 more books

    assert(s1.addBooks(['Math', 'Fantasy']) == True) #We can add books
    assert(s1.books == ['Math', 'Fantasy'])
    assert(s1.getSpace() == 3) #We can fit 3 more books

    assert(s1.addBooks(['Python']) == True) #True means we added successfully
    assert(s1.books == ['Math', 'Fantasy', 'Python'])
    assert(s1.getSpace() == 2) #Can fit 2 more.

    assert(s1.addBooks(['A', 'B', 'C', 'D', 'E']) == False) #False, not enough space!
    assert(s1.books == ['Math', 'Fantasy', 'Python']) #No books added

    s2 = Bookshelf(2) #Make a tiny bookshelf for two books
    assert(s2.books == [])

    #Move 'Math' and 'Python' from s1 to s2
    assert(s1.moveBooks(s2, ['Math', 'Python']) == True)
    assert(s1.books == ['Fantasy'])
    assert(s2.books == ['Math', 'Python'])
    assert((s1.getSpace() == 4) and (s2.getSpace() == 0))

    #If there isn't enough space, we don't move anything and return False
    assert(s1.moveBooks(s2, ['Fantasy']) == False)
    assert(s1.books == ['Fantasy'])
    assert(s2.books == ['Math', 'Python'])
```

3. Reasoning over Code [15pts]

Find an argument for the function `rc1(L)` that makes it return `True`. Place your answer (and nothing else) in the box to the right.

```
def rc1(L):  
    if (not isinstance(L, list)):  
        return False  
    A = []  
    B = []  
    while(L != []):  
        A.extend([L.pop(), L.pop(0)])  
        B = [L.pop(0), L.pop()] + B  
    return A == list(range(4, 8)) and A == B
```

L =

4. Reasoning over Code [15pts]

Find an argument for the function `rc2(L)` that makes it return `True`. Place your answer (and nothing else) in the box to the right.

```
def rc2(L):
    assert(isinstance(L, list) and (len(L) == 5))
    for i in range(-2, 2):
        assert(L[i] == L[i+1] + i)
    return (sum(L) == 20)
```

L =

5. Bonus CT (This problem is optional! Answer in the box to the right.) [3pts]

```
def bonusCt(L):
    def f(L): return [(-1)**(i%2>0)*L[i] for i in range(len(L))]
    L=f(L)
    return sum(L[L[1]:L[-2]:-2]) if (len(L)>4) else 42
print(bonusCt(list(range(11))))
```