

Name: \_\_\_\_\_ Section: \_\_\_\_\_ AndrewID: \_\_\_\_\_

15-112 Fall 2021 Quiz 4a

\* Up to 25 minutes. \* No calculators, no notes, no books, no computers. \* Show your work!

\* No recursion

**Code Tracing 1 [20pts]:** Indicate what the following code prints. Place your answers (and nothing else) in the box to the right of the code.

# Note: this prints 5 lines!

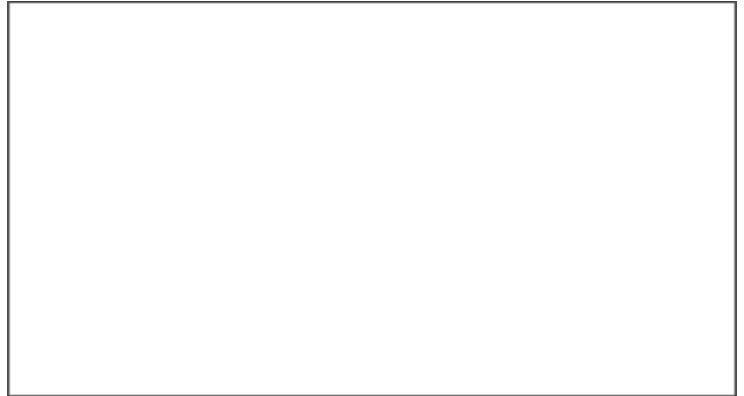
```
def ct1(L):  
    M, N = L, L+[1]  
    L[0] = 1  
    N[2] = 3  
    L += L  
    for x in [L, M, N]:  
        print(x)
```

```
L = [2,4]  
print(ct1(L))  
print(L)
```

**Code Tracing 2 [20pts]:** Indicate what the following code prints. Place your answers (and nothing else) in the box to the right of the code.

# Note: this prints 5 lines

```
def ct2(L):
    M = [ ]
    R = L[::-1]
    for k in R:
        if (k in L) and (k not in M):
            L.remove(k)
            M.append(k)
        else:
            M.append(k % 3)
            print(M.pop(0))
    return M
L = [2,4,6]*2
print(ct2(L))
print(L)
```



**Free Response 1: nondestructiveOddsBeforeEvens [35 pts]**

Write the function `nondestructiveOddsBeforeEvens(L)` that takes a possibly-empty list of integers `L` and nondestructively returns a new list with all the odds in `L` first, in the same order they appear in `L`, followed by all the evens in `L`, again in the same order they appear in `L`. So:

```
L = [1,2,3,4,5]:
```

```
assert(nondestructiveOddsBeforeEvens(L) == [1,3,5,2,4])
```

```
assert(L == [1,2,3,4,5]) #L does not change:
```

For full credit, your solution may not call the destructive version (`destructiveOddsBeforeEvens`) or apply very similar logic on a copy of `L`

## Free Response 2: destructiveOddsBeforeEvens [35 pts]

Write the function `destructiveOddsBeforeEvens(L)` that takes a possibly-empty list of integers `L` and destructively modifies `L` so that all the odds in `L` are first, in the same order they appear in the original list, followed by all the evens in the original list, again in the same order they appear in `L`. Your function should return `None`. So:

```
L = [1,2,3,4,5]
assert(destructiveOddsBeforeEvens(L) == None)
assert(L == [1,3,5,2,4])
```

## Bonus/Optional: Code Tracing [+2.5pts]

Indicate what this prints. Place your answer (and nothing else) in the box.

```
def bonusCt(L, M):
    while L != M:
        L,M = M,L
        M.append(L.pop(0))
        M.pop(0)
        L[-1] = sum(M)
    return L
print(bonusCt([1,2,3],[4,5,6]))
```