Quiz 4 Retake Free Response (70 pts)

Time Limit: **10 mins**

Name: _____ AndrewID: _____

Time limit: 10 mins

Free Response 1: nonmutatingReverseEvens (35 pts)

Note: This is the nonmutating version of the next exercise.

Write the function `nonmutatingReverseEvens(L)` that takes a list of integers `L` and returns a new list that is the same as `L`, but the order of the even numbers is reversed. The odd numbers should stay in the same positions they were originally.

For example, if `L = [3, 6, 0, 5, 2]`, then `nonmutatingReverseEvens(L)` should return `[3, 2, 0, 5, 6]`. The order of the even numbers (`6`, `0`, and `2`) are reversed in the result list. However, the odds (`3` and `5`) stay in their original positions.

You may not copy the input and use a mutating approach. Instead, you must build a new list from scratch.

You may only use material that was covered in units 1-4. Thus, do not use sets, dictionaries, OOP, or recursion.

Test cases:

```
def testNonmutatingReverseEvens():
    assert(nonmutatingReverseEvens([3, 6, 0, 5, 2]) == [3, 2, 0, 5, 6])
    assert(nonmutatingReverseEvens([0, 1, 0, 8]) == [8, 1, 0, 0])
    assert(nonmutatingReverseEvens([2, 1, 4]) == [4, 1, 2])
    assert(nonmutatingReverseEvens([4]) == [4])
    assert(nonmutatingReverseEvens([]) == [])

    # Verify that the function is non-mutating:
    L = [3, 6, 0, 5, 2]
    nonmutatingReverseEvens(L)
    assert(L == [3, 6, 0, 5, 2])
```

Write your answer below.

Free Response 2: mutatingReverseEvens (35 pts)

Note: This is the mutating version of the previous exercise.

Write the function `mutatingReverseEvens(L)` that takes a list of integers `L` and mutates `L` so the order of the even numbers is reversed. The odd numbers do not change. As with most mutating functions, this function should return `None`.

For example, if `L = [3, 6, 0, 5, 2]`, then after calling `mutatingReverseEvens(L)`, `L` should be `[3, 2, 0, 5, 6]`. The order of the even numbers (`6`, `0`, and `2`) are reversed. However, the odds (`3` and `5`) stay in their original positions.

You may not simply call your nonmutating function and overwrite all values in `L`.

You may only use material that was covered in units 1-4. Thus, do not use sets, dictionaries, OOP, or recursion.

Test cases:

```
def testMutatingReverseEvens():
    L = [3, 6, 0, 5, 2]
    assert(mutatingReverseEvens(L) == None)
    assert(L == [3, 2, 0, 5, 6])

    L = [0, 1, 0, 8]
    mutatingReverseEvens(L)
    assert(L == [8, 1, 0, 0])

    L = [2, 1, 4]
    mutatingReverseEvens(L)
    assert(L == [4, 1, 2])

    L = [4]
    mutatingReverseEvens(L)
    assert(L == [4])

    L = []
    mutatingReverseEvens(L)
    assert(L == [])
```

Write your answer below.