fullName:__________________ andrewID:________________ section:__

15-112 F24
Quiz4 version A (20 min)

Read these instructions carefully before starting:

1. Quiz versions are color-coded. You must have a different version (color) of this quiz than the students sitting to your left and right.
2. Stop writing and submit the entire quiz when instructed by the proctor.
    - Do not unstaple any pages.
    - You must submit the entire quiz with all pages intact.
3. Do not discuss the quiz with anyone else until after 4pm.
    - This applies to everyone, including students in either lecture.
4. Do not use your own scrap paper.
    - You should not need scrap paper, there is plenty of room for you on the quiz.
    - However, if you absolutely must use scrap paper, raise your hand and we will provide some. Then, you must write your andrew id clearly on the scrap paper, and hand in the scrap paper with your paper quiz. We will not grade anything on your scrap paper.
5. You may not ask questions during the quiz.
    - The one exception is for English-language clarifications.
    - If you are unsure how to interpret a problem, just take your best guess.
6. Do not use any concepts (including built-in functions) not covered in the notes through week 4.
    - Do not use 2d lists, dictionaries, sets, or recursion.
7. Do not hardcode your solutions.
    - We may test your code using additional test cases.
    - Hardcoding will receive zero points.
8. 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.
9. Good luck!

Code Tracing (CT) [40 pts, 4 pts each]
For each CT, indicate what the code prints
Place your answer (and nothing else) in the box below the code.

Reminder: `int(True)` evaluates to `1` and `int(False)` evaluates to 0

Note: every CT prints exactly one line.

```
def ct1(L):
    return L.append(2)
L = [ ]
R = ct1(L)
print(R, int(L == [ ]))
```

```
def ct2(L):
    L += [2]
    return L
L = [ ]
R = ct2(L)
print(R, int(L == [ ]))
```

```
def ct3(L):
    L = L + [2]
    return L
L = [ ]
R = ct3(L)
print(R, int(L == [ ]))
```

```
def ct4(L, M):
    sorted(M)
    L.sort()
L = [2, 1, 3]
M = [2, 1, 3]
ct4(L, M)
print(int(L == [1, 2, 3]), int(M == [1, 2, 3]))
```

```
def ct5(x, y):
    L = list(range(x, y))
    M = [v**2 for v in L if v % 10 == v]
    return M[::-1]
print(ct5(8, 12))
```

```
def ct6(L):
    M = L
    M.append(L.pop(0))
    return L + [0] + M
print(ct6([1, 2]))
```

```
def ct7(L):
    M = copy.copy(L)
    M.append(L.pop(0))
    return L + [0] + M
print(ct7([1, 2]))
```

```
def ct8(L):
    M = [L[-1-i]*2**i for i in range(len(L)) if i%2 == 1]
    return M
print(ct8([2, 4, 6, 1]))
```

```
def ct9(L):
    m = min([len(s) for s in L])
    L = [s[m:] for s in L]
    return [int(v) for v in L if bool(v)]
print(ct9(['123', '456', '78', '90', '9876']))
```

```
def ct10(L):
    for v in range(5):
        for _ in range(v):
            if v in L:
                L.remove(v)
    return '-'.join([str(v) for v in L])
print(ct10([0, 1, 2, 3] * 3))
```

Multiple Choice (MC) [25 pts, 5 pts each]

For each MC, fill in the bubble next to the correct answer. Each question has only one answer.

MC1: If `L = list(range(1, 6))`, which of the following is NOT equal to 3?

- ◯ `L[0] + L[1]`
- ◯ `L[2]`
- ◯ `len(L[:3])`
- ◯ `L[-3]`
- ◯ `L[-2]`

MC2: Which of the following is NOT true?

- ◯ If `L` is an alias of `M`, then `L` is `M` must be `True`.
- ◯ If `L` is a copy (and not an alias) of `M`, then `L` is not `M` must be `True`.
- ◯ If `L == M` is `True`, then `L is M` must be `True`.
- ◯ If `L is M` is True, then `L == M` must be `True`.

MC3: For what list L will the following code NOT print 'yes'?

```
M = sorted(L, reverse=True)
N = list(reversed(M))
print('yes' if L == N else 'no')
```

- ◯ `L = [3,2,1]`
- ◯ `L = [5]`
- ◯ `L = [1,2,3]`
- ◯ `L = list(range(32))`

MC4: Which of the following is False?

- ○ For a multiline string `s` that does not end in a newline, if `L = s.splitlines()`, then `s == '\n'.join(L)` must be True.
- ○ For any string `s` that does not end in a newline, `s.split('\n') == s.splitlines()` is always True.
- ○ If `L` is a list of names, then `','.join(L)` will be a comma-separated string of those names.
- ○ For any list `L`, if `s = '-'.join(L)`, then `'-' in s` must be True.

MC5: Which of the following is False?

- ○ If `t = ( [1], [2] )`, we cannot set `t[0] = 3`, but we can set `t[0][0] = 3`.
- ○ `((1, 2) != [1, 2])` is True
- ○ `(4,)` is a singleton tuple but `(4)` is not a tuple at all.
- ○ We cannot use += with tuples, because tuples are immutable.

Free Response (FR): oddsAndEvens(L) [35 pts]

Write the function `oddsAndEvens(L)` that takes a list `L` of ints and mutates `L` by removing all the odd values. The function should not return `None`. Instead, the function should return a list of the odd values it removed in their original order.

Here are some test cases for you:

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

L = [ -4, -3, -2, -1, -2, -3, -4 ]
assert(oddsAndEvens(L) == [ -3, -1, -3 ])
assert(L == [ -4, -2, -2, -4 ])

L = [ 1, 3, 1, 5 ]
assert(oddsAndEvens(L) == [ 1, 3, 1, 5 ])
assert(L == [ ])

L = [ 2 ]
assert(oddsAndEvens(L) == [ ])
assert(L == [ 2 ])

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

Begin your FR answer on the next page.

Begin your answer to the FR here:

Bonus Code Tracing (BonusCT) [Optional, 4 pts, 2 pts each]
Bonus problems are not required.
For each CT, indicate what the code prints
Place your answer (and nothing else) in the box below the code.

BonusCT1:

```
def bonusCt1(L):
    x, y, L = 0, 0, L[:]
    while L:
        z = 0
        for i in range(len(L)-1, -1, -1):
            L[i] -= 1
            if not L[i]: L.pop(i); z += 1
        if x: y += 1
        if z: x += 1
    return y
print(bonusCt1([8, 24, 18, 37, 26]))
```

BonusCT2:

```
def bonusCt2(L):
    def f(L): return (f(L[:len(L)//2]) +
                      [sum(L)] +
                      f(L[len(L)//2:])
                      if len(L)>1 else L)
    return f(f(L))
print(bonusCt2([1, 3]))
```