Name:__________________________________ Section:___ Andrew Id: ____________________ a

15-112 Fall 2019 Quiz 4a

*** Up to 33 minutes. No calculators, no notes, no books, no computers. * Show your work!**
*** No recursion**

1. **Free Response: shortenLongRuns(L, k)** [25 pts]
   Write the **non-destructive function shortenLongRuns(L, k)** that takes a list L and a positive integer k, and **non-destructively** returns a similar list, only without any run of k consecutive equal values in L. This means that any values that would otherwise produce a consecutive run of k elements are not present.

   For example:
   ```
       assert(shortenLongRuns([2, 3, 5, 5, 5, 3], 2) == [2, 3, 5, 3])
       assert(shortenLongRuns([2, 3, 5, 5, 5, 3], 3) == [2, 3, 5, 5, 3])
   ```

   Note: your function may not just create a copy of L and call the destructive version of this function (below) on that copy and return it. Instead, you must directly construct the result here.

2. **Free Response: Destructive shortenLongRuns(L, k)** [25 pts]:
Here we will rewrite the previous function to be **destructive.** This version must not just call the non-destructive version from above and then clear and replace the values in the original list. Instead, you must traverse the list once and make the changes to the list as you go.

With that in mind, write the **destructive function shortenLongRuns(L, k)** that takes a list L and a positive integer k, and destructively modifies L by removing any values in L that would otherwise produce a run of k consecutive equal values in L.

For example:
```
    L = [2, 3, 5, 5, 5, 3]
    shortenLongRuns(L, 2)
    assert(L == [2, 3, 5, 3])
```

And:
```
    L = [2, 3, 5, 5, 5, 3]
    shortenLongRuns(L, 3)
    assert(L == [2, 3, 5, 5, 3])
```

3. **Free Response: SayHi class** [20 pts]
   Write the **class SayHi** so that the following test function passes (and without hardcoding any test cases):

```
def testSayHiClass():
    print('Testing sayHiClass...', end='')
    harry = SayHi('A')
    ron = SayHi('B')
    hermione = SayHi('C')
    hermione.sayHi(ron)
    assert(harry.saidHiList() == [ ])
    assert(ron.saidHiList() == [ hermione ])
    assert(hermione.saidHiList() == [ ])
    hermione.sayHi(ron) # include duplicates
    harry.sayHi(ron) # and list in the order they say hi, so...
    ron.sayHi(harry)
    assert(harry.saidHiList() == [ ron ])
    assert(ron.saidHiList() == [ hermione, hermione, harry ])
    assert(hermione.saidHiList() == [ ])

    # Finally, a similar method returns the names, not the objects:
    assert(harry.saidHiNames() == [ 'B' ])
    assert(ron.saidHiNames() == [ 'C', 'C', 'A' ])
    assert(hermione.saidHiNames() == [ ])
    print('Passed!')
```

4. **Code Tracing CT1** [10 pts]:
   Indicate what this prints. Place your answer (and nothing else) in the box next to the code.

```
def ct1(s):
    t = s
    u = copy.copy(s)
    s += 'b'
    return s, t, u

for value in ct1('a'):
    print(value)
for value in ct1(['a']):
    print(value)
```

5. **Code Tracing CT2** [10 pts]:
   Indicate what this prints. Place your answer (and nothing else) in the box under the code.

```
def ct2(s, L):
    i = 0
    M = [ ]
    while (i < len(L)):
        if (not L[i].isalpha()):
            i += 1
        else:
            u, t, s = L.pop(i), s[0], s[1:]
            if (u == t):
                M.append(i)
    return M

L = ['a', '2', 'bc', '#', 'd', '4']
s = 'abde'
print(ct2(s, L))
print('-'.join(L))
```

6. **Reasoning Over Code** [10 pts]:
Find arguments for the following functions that makes them return True. Place your answers (and nothing else) in the box under the code:

```
def rc1(L):
    if ((L == [ ]) or (min(L) != 5)): return False
    M = [ ]
    for i in range(1, len(L)):
        d = L[i] - L[i-1]
        if (d < 1) or (d in M):
            return False
        M.append(d)
    return ((len(M) == 3) and
            (M[0] == 1) and
            (sum(M) == 6) and
            (M != sorted(M)))
```

L =

7. **Bonus/Optional: Code Tracing** [2.5 pts Each]
Indicate what this prints. Very clearly circle your answer (and nothing else):

```
def bonusCt1():
    n = 1
    while True:
        n, L, i = n+1, list(range(n)), 0
        while (i+1 < len(L)):
            L[i] += L.pop(i+1)
            i += 1
        if (L[-1] > 40):
            return sum(L)
print(bonusCt1())
```

```
def bonusCt2(L):
    return sum([int(d) for d in
                       str([str(c*2)*2 for c in L*2]) if d.isdigit()])
print(bonusCt2([4,5,6]))
```