

fullName:_____ andrewID:_____ section:_____

15-112 N25

Quiz3

This is the paper version of Quiz3. You **must write your name on this paper and hand this back** in immediately after the assessment, even if you do not use it to write your answers. If we do not receive it immediately, you will receive a zero on the assessment.

Quiz3 is located in section 3.15 of CS Academy (the end of Unit 3), and contains two parts. You must do Part 1 first, and you must submit it before viewing Part 2. If you cannot see the submit button, call over a TA or Mike and let us know your andrewID.

We will not grade anything you write on these pages unless you initial the box below:

_____ **Write your initials here if and only if you wish for us to grade what you have written on this paper *instead of* any answers submitted in CS Academy.** Leave the space blank if you wish for us to grade what is written in CS Academy (the default, recommended option). Note: Even if you write your answers on paper, you may not run any code related to the Part 1 problems (CTs).

You may not view any other notes, prior work, websites or resources, including any form of AI. You may not communicate with anyone else except for TAs or faculty during the assessment. All syllabus policies apply.

Some students will take this at a different time with testing accommodations. As such, you may not discuss this test with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use sets, dictionaries, recursion, or anything else disallowed in the original problem.

Do not open this or look inside (even briefly) before we instruct you to begin. Close it once you are done.

Part 1 [30pts total]: Code Tracing

CT1[15 pts]:

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

We strongly recommend making a box-and-arrow diagram for list CTs.

```
import copy
def ct(L):
    A = L
    B = copy.copy(L)
    x = L.pop()
    B += ['z']
    print('L =',L)
    print('A =',A)
    print('B =',B)
    for i in range(len(L)):
        if type(L[i]) == int and L[i] > x:
            x = L[i]
    return x

print(ct(['x', 20, 'y', 5]))
```

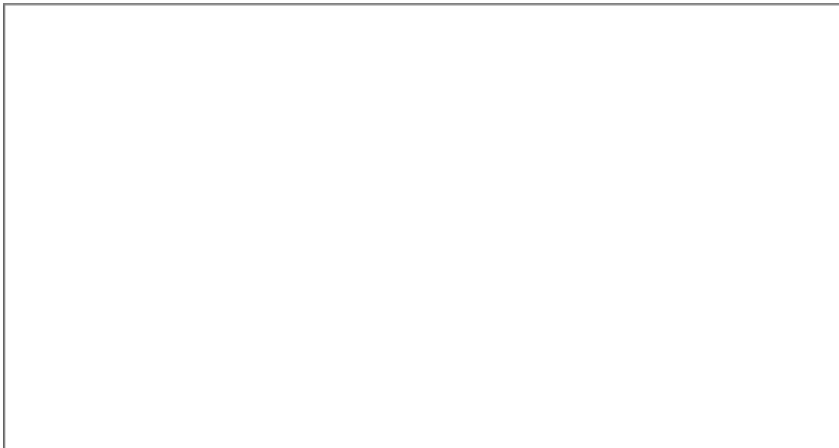
CT2[15 pts]:

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

We strongly recommend making a box-and-arrow diagram for list CTs.

```
def ct(L):
    i = 0
    M = [False]
    N = L
    while i < len(L):
        if type(L[i]) == int:
            M.append(L.pop(i))
            N = N[1:]
        i += 1
    return M + N
L = ['abc', 123, 456, True, 'xyz', 7]

print(ct(L))
print(L)
```



Part 2[70pts total]: Free Response

Your functions should work generally for the kinds of inputs specified in the problem statement, and we may test your code using additional test cases. We will manually grade both of these problems for partial credit if you do not pass all the test cases.

FR1[30pts]: `removeAutoIndexes(L)`

Background: For a list `L` and non-negative index `i`, we will say that `L[i]` is an "auto-index" if `L[i] == i`. For example, in the list `L = [4, 1, 0, 3]`, we see that `L[1] == 1` and `L[3] == 3`, so `L[1]` and `L[3]` are both auto-indexes.

With this in mind, write the **mutating** function `removeAutoIndexes(L)` that takes a possibly-empty list of integers `L` and **mutates** `L` so that all the auto-indexes in `L` are removed. The function should also return the number of removed values.

Your function **must meaningfully mutate L!** Non-mutating functions will not receive more than half credit.

Here are some test cases:

```
L = [4, 1, 0, 3] # The original value of L
assert(removeAutoIndexes(L) == 2) # Return how many vals removed
assert(L == [4, 0]) # Test that L has been mutated

L = [1, 4, 3, 0]
assert(removeAutoIndexes(L) == 0)
assert(L == [1, 4, 3, 0])

L = [0, 1, 2, 3, 4, 5]
assert(removeAutoIndexes(L) == 6)
assert(L == [ ])

L = [ ]
assert(removeAutoIndexes(L) == 0)
assert(L == [ ])

L = [3, 5, 9, 3, 4, 5, 0]
assert(removeAutoIndexes(L) == 3) # Only remove autoIndexes
assert(L == [3, 5, 9, 0]) # Make sure to remove the right 3 and 5!
```

Note that we remove auto-indexes based on their initial indices in `L`. We do *not* remove integers that become auto-indexes later. For example:

```
L = [0, 0, 1]
assert(removeAutoIndexes(L) == 1) # Remove only the 0 at L[0]
assert(L == [0, 1]) # These ints were not originally auto-indexes
```

Define removeAutoIndexes(L) below:

FR2[40 pts]: movingDot Animation

Hint: This writeup is written so that you can implement the steps one at a time in order, but read all of the steps first. Make sure to interact with the solution canvas, and test your code in between every step! **In all animation problems, your code must adhere to MVC.**

Write an app that has the following properties:

- It starts with a green circle of radius 50 drawn in the center of the canvas.
- A number is drawn at coordinates (50, 50) indicating the radius of the circle.
- If the mouse is pressed inside the dot (or exactly on its border), the radius increases by 10 pixels. If the radius reaches 150 pixels, it no longer increases in size.
- If the mouse is pressed outside the dot, the radius decreases by 10 pixels. If the radius reaches 20 pixels, it no longer decreases in size
- Initially, the app is paused. Pressing 'p' pauses/unpauses the app.
 - When the app is unpaused, 5 times per second, the dot moves 12 pixels downward. If any part of the dot goes off the bottom edge of the canvas, that part wraps around so that it begins to re-enter at the top edge of the canvas.
 - When the app is paused, the dot does not move.
 - **Note:** Pausing does not affect mouse presses. Pausing only affects the downward movement.
 - **Note:** Make sure your mouse presses still work in the wrapped around dot!

Note: You may assume the canvas is 400x400

Write your animation on the following page

```
# Begin the movingDot animation on this page:  
# You may define additional helper functions if you wish!
```

```
from cmu_graphics import *
```

```
def onAppStart(app):
```

```
def distance(x1, y1, x2, y2): #You may use this function if you wish  
    return ((x1 - x2)**2 + (y1 - y2)**2)**0.5
```

```
def redrawAll(app):
```

```
def onMousePress(app, mouseX, mouseY):
```

```
def onKeyPress(app, key):
```



```
def onStep(app):
```

#Extra space for FR2 helper functions if needed

```
def main():  
    runApp()  
main()
```