Name: ______________________ Recitation: ______ Andrew Id: ______________

15-112 Fall 2018 Quiz 8

Up to 25 minutes. No calculators, no notes, no books, no computers. Show your work!
Do not use recursion on this quiz.

1. (15 points) **Reasoning Over Code**: Find an argument for the following program that makes it return `True`. Place your answer (and nothing else) in the box under the code.

```
def ROC(lst):
    d={}
    while len(lst)>0:
        key=lst.pop(0)
        item=lst.pop(0)
        d[key]=item

    if False in d.keys():
        return False

    d[1].remove(1)

    if d['cat']==3:
        d[False]=d['dog']
    else:
        d['cat']=True

    assert(len(d.values())==4)

    if d[1]=={'bird'}:
        return d[False]
    return False
```

2. (15 points) **Short Answer:** Briefly explain why mutable data types can't be used as items in a set or as keys of a dictionary.

Built-in Big-O Runtimes	
General	
`isinstance(item, type)`	$O(1)$
`len(item)`	$O(1)$
`item[i]`	$O(1)$
Strings	
`c in s`	$O(N)$
Lists	
`lst.append(item)`	$O(1)$
`lst[i:j:k]`	$O((j-i)/k)$
`lst.insert(i, item)`	$O(N)$
`item in lst`	$O(N)$
`min(lst) / max(lst)`	$O(N)$
`lst.reverse()`	$O(N)$
`lst.sort()`	$O(NlogN)$

3. (25 points) **Short Answer:** For each of the three functions shown below, write next to each line of the function either the Big-O runtime of the line or the number of times the line loops. Then write the total Big-O runtime of the function in terms of N in the box to the right of the code.

```
def foo(a):  # a is an N x N 2D-list          # Big-O
    N = len(a)                                #______
    L = list(range(N, N**2, 42))              #______
    for r in range(N):                        #______
        for c in range(N):                    #______
            g = a[r][c]                       #______
            d = binarySearch(L, g)            #______
            if g == N + 42:                   #______
                L.pop(L.index(g))             #______
    return a[0]*N                             #______
```

```
def bar(a):  # a is a list of ints            # Big-O
    N = len(a)                                #______
    for startIndex in range(N):               #______
        minIndex = startIndex                 #______
        for i in range(startIndex+1, N):      #______
            if (a[i] < a[minIndex]):          #______
                minIndex = i                  #______
        temp=a[startIndex]                    #______
        a[startIndex]=a[minIndex]             #______
        a[minIndex]=temp                      #______
```

4. (25 points) **Free Response:** Write the function `hasSquarePair(lst)` that takes a list of positive integers and returns `True` if the list contains an integer whose square is also in the list. So [4,3,6,12,9] returns `True` because 3^2 is 9, which is also present in the list. Likewise, the list [4,7,5,22,30] returns `False` because the list contains no integer whose squared value is also in the list. **Your function must have an efficiency of O(N) or better!** Here, N is the length of the list. Write next to each line of the function either the Big-O runtime of the line or the number of times the line loops. Then **clearly** write the Big-O runtime of the function in terms of N next to the function name, and circle it.

5. (20 points) **Free Response:** Write the animation functions `init(d)`, `timerFired(d)`, and `redrawAll(c,d)` to meet the requirements below. Assume that the run function is defined as in the notes (with a timerDelay of 100), and we do not need `mousePressed(e,d)` or `keyPressed(e,d)`.

   The animation draws a 40x40 square in the center of the canvas, filled with some color, and a number in the center of the square. The animation has the following properties:

   - The number counts down by 1 every second, starting at 60.
   - Once the number reaches zero, it resets to 60.
   - Every 100ms, the box should move vertically by 3 pixels. The starting direction is up.
   - When the edge of the box hits the edge of the canvas, it should bounce and change it's direction appropriately, i.e. if the top of the box hits the top of the canvas, it begins moving downward, and if the bottom of the box hits the bottom of the canvas, it begins moving upward.
   - The number should remain in the center of the square at all times!

   Hint: You should use d, e, and c in place of data, event, and canvas to save time!