Name: _____ Andrew Id: _____

15-112 Spring 2019 Quiz 8

Up to 25 minutes. No calculators, no notes, no books, no computers. Show your work!

1. (30 points) **Free Response:** Write the function `cheapestProducts(priceList)` which, given a list of tuples each containing the product name, store name, and price of a product, returns a dictionary mapping product names to a tuple containing the store and price for the lowest price found for the item. In the event of a tie, either store may be used.

   Consider the following example. If

   ```
   productList = [ ("X-Box", "LuLu", 1200),
                   ("iPhone XR", "Carrefour", 3000),
                   ("Washer", "SafariMart" , 799),
                   ("iPhone XR", "Al Anees" , 2870),
                   ("Washer", "LuLu", 899)
                 ]
   ```

   then `cheapestProducts(priceList)` returns a dictionary containing...

   ```
   {'X-Box': (1200, 'LuLu'), 'iPhone XR': (2870, 'Al Anees'), 'Washer': (799, 'SafariMart')}
   ```

   To get full credit, your solution must run in $O(N)$ time, where N = `len(productList)`.

2. (20 points) **Short Answer:** For both of the two functions shown below, write next to each line of the function either the Big-O runtime of the line. (For loop lines, instead write 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. You must write in the Big-Os/number of loops of each line to get full credit.

```
def f1(n): # n is an int, N = n                    # Big-O
    ret = []                                       #______
    for i in range(n):                             #______
        for j in range(i+1,n):                     #______
            ret.append(i+j)                        #______
    return ret                                     #______
```

```
def f2(L): # L contains N items                    # Big-O
    allowedWords = ["surround", "friend", "stop"]  #______
    newList = []                                   #______
    for w in L:                                    #______
        if w in allowedWords:                      #______
            newList.append(w)                      #______
    for w in allowedWords:                         #______
        if w in L:                                 #______
            L.remove(w)                            #______
        L.sort()                                   #______
    L.extend(newList)                              #______
    return L                                       #______
```

Big-O Runtimes for a List `len(lst) == N`	
`lst.append(item)`	$O(1)$
`lst.remove(item)`	$O(N)$
`item in lst`	$O(N)$
`lst.sort()`	$O(N \log N)$
`lst.extend(lst2)`	$O(k)$ where `len(lst2) == k`

3. (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

4. **Short Answer:** The following function returns the number of elements in L that also have their square in L. This function is less efficient than it could be. Briefly describe how to change the function to make it more efficient without completely changing the algorithm. Your answer must change the function family of the program's runtime.

```
def countSquares(L):
    count = 0
    for item in L:
        square = item**2
        if item != square and square in L:
            count += 1
    return count
```

   (a) (5 points) What is the Big-O efficiency of this function?

   (b) (10 points) Briefly describe how to change the function to make it more efficient without completely changing the algorithm. Your answer must change the function family of the program's runtime. Also state the Big-O efficiency the function with your changes.

5. (20 points) **Code Tracing:** Indicate what the following program prints. Place your answer (and nothing else) in the box under the code.

```
def ct(L):
    a = dict()
    for b in L:
        if b[0] not in a:
            a[b[0]] = set()
        a[b[0]].add(b)

    k = a.keys()
    o = sorted(list(k))
    print(len(k))
    print(o)
    print(a[o[-1]])

m = ["surround", "friend", "stop", "plane", "glossy", "bouncy", "puncture",
     "mate", "plant", "chin", "powerful", "cars"]
ct(m)
```