Here's some practice as a Ch 5 Review (Use anything that you've learned in Ch 4 though.) In particular, this problems will help you review:

slicing
lambda
list comprehensions
sorting x
searching for items in a list
creating, adding to a list

Though I do not state it in each problem, you should call your function in a variety of ways to test it.

1. Ask the user for a word and a start/end letter. Return the portion of the word between the first occurrence of the start letter and the last occurrence of the end letter. Using slicing to get the desired subword.

   Recall: what function gives you the position of the FIRST occurrence of an item? (The answer is in the posted 5C notes if you don't know.)

2. We can remove duplicates in a list in many ways. Here's two more ways.

   Say values = [0, 0, -10, 15, 13, 12, 0, -10, 15, 13, 12, -10, 13, 12]

   (a) Create a function removeDuplicates that takes in a list and returns a list containing a new list of only the unique values. Call the function on the values list above.

```
def removeDuplicates(myList):
      #create an empty list called unique here.
      #We'll now add items to unique so that we never add an item twice.
      #Here's some hints how.
      for item in myList:
            #if the item isn't already in the unique, then put it in there.
```

   (b) We can also use list comprehensions to get a list of unique items. This is more mathematical, but smooth.

   - First go to the posted notes from Ch5A and remind yourself what "enumerate" does and then fill in the first blank below.
   - Finally, fill in the second blank below. HINT: If the number 15 appears at 3 and 8 like in the list below, then you'd only want to add it to the uniqueList when you analyze it at position 3.

```
values = [0, 0, -10, 15, 13, 12, 0, -10, 15, 13, 12, -10, 13, 12]
uniqueList = [item for _____________   in enumerate(values) if  _____________]
```

3. Insert 20 random letters into the range a-f into a list. Perform the following tasks, displaying your results.

   (a) Sort the list in ascending order
   (b) Sort the list in descending order
   (c) Sort the unique values in ascending order

   HINT: One way to get random letters from 'abcdef' is to just pick random indices from this string:
   word = 'abcdef'

4. Say you have a tuple of students and grades, like below.

   [('Ann', 82), ('Mary', 75), ('Terry', 85), ('Clare', 94), ('Kitty', 87), ('Mickey', 77), ('Aggie', 72)]

   (a) Use list comprehensions to list the GRADES of everyone who got in the B range (80-89)

   (b) Use filter/map to list the NAMES of everyone (in all capital letters) who got in the B range (80-89)

5. Sum the squares of the odd numbers from 40 to 60 using the following. FYI: The correct answer is 25330.

   (a) list comprehensions

   (b) generator expressions

   (c) filter/maps with lambda expressions

6. Given a string of letters, replace every occurrence of a vowel with a capital V. Do this by using filtering/mapping.

7. Given a list of numbers (such as our favorite array below), print a list of each value and its frequency. Do this without removing duplicates. (Hint: just find the min/max of the values.)

```
dailyChanges = [1, -4, 5, -8, -3, 14, -13, 8, 4, -4, -9, -11, -3, 11, -9, 5,
    -11, 7, 5, -15, 6, 11, -12, -2, 0, -4, 5, 3, 2, 11, 5, -1, 5,
    3, -1, 2, -3, 0, 1, -7, -11, 0, 7, 0, -4, -5, 12, -1, 3, 2,
    -1, -2, 2, -10, -10, 8, 7, -6, -9, -1, 1, 10, -5, 4, 5, 1,
    -10, -3, -2, 5, 1, -4, 1, 1, 3, 9, 6, -1, -8, 5, -27, 10, -1,
    -8, -3, -5, 6, 14, 13, -25, -8, -3, 14, 2, 7, -3, 4, 1, -10,
    4, 7, -5, -7, -1, -2, -7, -5, 3, 16, 8, -2, -14, -2, 7, -4,
    -4, 12, 0, -10, -5, -2, 1, -6, -9, 14, 0, 1, -5, -9, -3, 8,
    -2, 17, -4, -4, 0, -2, -16, 8, 6, -10, -1, 13, 0, -6, 3, 11,
    -14, -6, -3]
```

If you allow 0's, then the answer would be that shown on the next page.

-27: 1
-26: 0
-25: 1
-24: 0
-23: 0
-22: 0
-21: 0
-20: 0
-19: 0
-18: 0
-17: 0
-16: 1
-15: 1
-14: 2
-13: 1
-12: 1
-11: 3
-10: 6
-9: 5
-8: 4
-7: 3
-6: 4
-5: 7
-4: 9
-3: 9
-2: 9
-1: 9
0: 8
1: 10
2: 5
3: 6
4: 4
5: 9
6: 4
7: 6
8: 5
9: 1
10: 2
11: 4
12: 2
13: 2
14: 4
15: 0
16: 1
17: 1