Work with your neighbor. (This will be graded for participation only.)

1. The following code to remove the last element in a linked list does not work for all linked lists:

```
def remove_last(self):
    if self._head == None:    #the list is empty
        return None

    curr = self._head
    while curr._next != None:
        prev = curr
        curr = curr._next
    prev._next = None
    return curr
```

   a) Give an example linked list that causes this code to crash with an exception.

   b) Write a correct version of the code.

2. This function attempts to verify that a tree is a Binary Search Tree (BST). Unfortunately, it has a few bugs. (Note: an empty tree and a leaf are both BSTs.)

```
def is_BST(root):
    if root is None:
        return True
    if root._left is None and root._right is None:
        return True
    if root._left._val > root._val or
                root._right._val < root._val:
        return False
    return is_BST(root._left) and is_BST(root._right)
```

a) The function errors out on many trees. Give an example of a tree that makes this function crash. Then fix that bug and write the fixed version below:

b) Does your fixed code work for all BSTs? If not, give an example of a tree for which your improved function will return an incorrect answer. (I.e., it returns `True` even if the tree is not a BST).

c) Explain why your function does not work for your example tree.

d) How would you fix this problem? You don't need to write the code, just explain in a couple of sentences what you need to do for a correct solution.

3. Answer the questions for each of the code snippets below. Which value(s) in this code should you print out in order to debug this? Why is the code sometimes failing?

   a) code snippet 1

```
data1 = …a Python list…
data2 = set(data1)
assert len(data1) == len(data2)
```

   Why is the code sometimes failing with an assertion error? Which value(s) in this code should you print out in order to debug this?

   b) code snippet 2

```
data1 = …a Python list…
data2 = sorted(data1)
assert data2[0] <= data2[1]
```

   Why is the code sometimes failing with an error? Which value(s) in this code should you print out in order to debug this?

For problems 4, 5, and 6 on the next pages, you can download `ICA-38.py` from the class website if you'd like to run the code.

4. The calculation for BMI is weight in kilograms divided by the square of the height in meters. The function below computes the BMI given a weight and a height:

```
def calculate_bmi(weight, height):
    return weight / (height ** 2)
```

The following code produces the same BMI for the pairs of weights and heights in the `patients` list:

```
patients = [(70, 1.8), (80, 1.9), (150, 1.7)]
for patient in patients:
    weight, height = patients[0]
    bmi = calculate_bmi(height, weight)
    print("Patient's BMI is:", bmi)
```

Output:

```
Patient's BMI is: 0.00036734693877551024
Patient's BMI is: 0.00036734693877551024
Patient's BMI is: 0.00036734693877551024
```

What would you print out to find the bug? What is the bug?

5. The following function takes a list of strings `alist` and returns a list of all of the strings in `alist` where the first character matches the last character:

```
def first_matches_last(alist):
    match_list = []
    for elem in alist:
        if elem[0] == elem[-1]:
            match_list.append(elem)

    return match_list
```

a) Give an example list that causes this code to fail with an exception.

b) Write a correct version of the code.

6. The class `Word` defines a word object and implements `__eq__()` to return `True` if two `Word` objects are anagrams and `False` otherwise:

```
class Word:
    def __init__(self, word):
        self._word = word

    def __eq__(self, other):
        if len(self._word) == len(other._word):
            for letter in self._word.lower():
                if letter not in other._word.lower():
                    return False
        return True
```

The method works for the following examples:

```
w1 = Word("post")
w2 = Word("stop")
print(w1 == w2)

w1 = Word("keep")
w2 = Word("peep")
print(w1 == w2)
```

e) Does this code work for all words? If not, give an example of two words that give an incorrect answer for equality.

f) Rewrite the method to fix any bugs you have found.

Final exam review (various questions on complexity)

7. In lecture we covered a simple memory model and showed how Python lists are laid out in memory.
   (a) Based on that discussion, what is the fundamental reason that indexing into a Python list and appending to a Python list are both O(1)?

   (b) Why is inserting into a Python list O(n)?

8. Write a function called `func(s)`, where `s` is a string, such that the complexity of `func` is O(n). Your function can have any purpose; just make it O(n).

9. What is the complexity of searching for a key in a hash table?