Extra Practice Questions – Post Midterm Material

V2 – 2017-03-22: Corrected solution to Q1.a. Extend adds each character as a separate entry only.

V3 – 2017-04-03: Corrected import in Q16. Clarified in Q20 that the description is sortED() not just sort(), and clarified solution.

Note: These questions are intended only as additional exercises that you can do in preparation for the final exam -- they primarily focus on the material that was covered after the midterm. The format, difficulty, coverage and length of the final exam may differ.

If you have any questions about these questions or the solutions, please post to Piazza.

Question 1

a. Complete the table below. If there is no output printed you can leave that box blank.

<table>
<thead>
<tr>
<th>Commands in the console</th>
<th>Output printed</th>
<th>Value of x after commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt;&gt; x = [1, (23,)]</td>
<td></td>
<td>[1, (23,)]</td>
</tr>
<tr>
<td>&gt;&gt;&gt; x.append('simon')</td>
<td></td>
<td>[1, (23,), 'simon']</td>
</tr>
<tr>
<td>&gt;&gt;&gt; x = [1, (23,)]</td>
<td></td>
<td>[1, (23,), 's', 'i', 'm', 'o', 'n']</td>
</tr>
<tr>
<td>&gt;&gt;&gt; x.extend('simon')</td>
<td></td>
<td>[1, 23, 99]</td>
</tr>
<tr>
<td>&gt;&gt;&gt; x + [99]</td>
<td></td>
<td>[1, 23 ]</td>
</tr>
</tbody>
</table>

Note: In the first two parts, the comma inside the parentheses in "(23,)" indicates that the second element in x is a tuple (containing a single element which is an integer). In the final part, there is no comma so the second element in the list is an integer, not a tuple.

b. Give one example of a process or task which would be easier to implement in Scratch than Python, and briefly explain why it is easier in Scratch. Give one example of a process or task which would be easier to implement in Python than Scratch, and briefly explain why it is easier in Python.

Any task that involves animation of many agents is easier implemented in Scratch than Python. It is more difficult to create graphics and animate them in Python. It is also easier to have multiple scripts running at the same time.

Any process that involves extensive calculations or reading/storing data to files is much easier implemented in Python than Scratch. Scratch does not support files and its operations with numerical values are limited.

c. Suppose a byte in the computer memory contains the following value 01110101. What kind of data can this value represent? Who decides what this value represents?

This value can represent any type of data: can be part of an integer or floating point number, can be part
of a string, part of a picture, etc.
The program that uses this byte will determine what this sequence of 0s and 1s represents and use it accordingly.

a. The difference between an alias and a copy can cause problems with lists, but not strings. Why?

Lists are mutable and strings are immutable.
**Question 2**
Consider the following Python program

```python
def bar1(y):
    x = len(y) - 1
    while(x >= 0):
        if(x % 2 == 0):
            y[x] = ''
        x = x - 1

# Main
x = 5
names = ['Jones', 'Tutu', 'Aziz', 'Butcher', 'Malkin', 'Chan']
result = bar1(names)
```

a. Draw the state of memory after the program has finished executing. You may use the simplified memory model notation we use in lecture, or notation similar to that used in the Python visualizer, as long as your drawing is clear and correct. Marks will be deducted for any extra variables/frames.

```
Global Frame
x → 5

names → ['', 'Tutu', '', 'Butcher', '', 'Chan']

result → None  (the function does not return any value, so it returns None)
```
Question 3
Consider the following Python program

```python
def bar2(y):
    x = len(y) - 1
    z = []
    while(x >= 0):
        if(x % 2 == 0):
            z.append(y[x])
        x -= 1
    return z

# Main
x = 5
names = [ 'Jones', 'Tutu', 'Aziz', 'Butcher', 'Malkin', 'Chan' ]
result = bar2(names)
```

a. Draw the state of memory just **before** the call to bar2 returns. You may use the simplified memory model notation we use in lecture, or notation similar to that used in the Python visualizer, as long as your drawing is clear and correct. Marks will be deducted for any extra variablesrames.

```
Global Frame                                           Data
 x ----------> 5

names ----------> ['Jones', 'Tutu', 'Aziz', 'Butcher', 'Malkin', 'Chan']

                        -1

bar2() local Frame
 y      z

x
```

**Note – This solution is drawn a little differently from some of the other memory model solutions in this package in order to clearly show aliases.**
b. Draw the state of memory after the program has finished executing. You may use the simplified memory model notation we use in lecture, or notation similar to that used in the Python visualizer, as long as your drawing is clear and correct. Marks will be deducted for any extra variables/frames.

Global Frame
x → 5

names → ['Jones', 'Tutu', 'Aziz', 'Butcher', 'Malkin', 'Chan']

result → ['Malkin', 'Aziz', 'Jones']
Question 4
Consider the following Python program

```python
extensions = { 'Jones' : 547, 'Aziz' : 352, 'Butcher' : 978,
              'Malkin' : 177, 'Chan' : 444, 'Tutu' : 277 }
calls = [ 'Aaronson', 'Freeman', 'Aziz', 'Sastry', 'Chen',
          'Jones', 'Tutu', 'Luongo' ]
s = []
n = 0
for p in calls:
    if p in extensions:
        s.append(extensions[p])
    else:
        s.append(None)
    n += 1
```

a. Draw the values of extensions, calls, s and n. You may use the simplified memory model notation we use in lecture, or notation similar to that used in the Python visualizer, as long as your drawing is clear and correct.

**Global Frame**

- **Extensions** → {'Jones': 547, 'Aziz': 352, 'Butcher': 978,
  'Malkin': 177, 'Chan': 444, 'Tutu': 277 }
- **Calls** → ['Aaronson', 'Freeman', 'Aziz', 'Sastry', 'Chen',
  'Jones', 'Tutu', 'Luongo']
- **s** → [None, None, 352, None, None, 547, 277, None]
- **n** → 5
Question 5
Consider the following Python program

```python
def foo(a, b, c):
    if a < b:
        c = a + b
    else:
        c = a - b
    return c
def spam(c, b, a = 3):
    b = a + c
    d = a + b + c
    return d
a = 2
b = 1
c = 0
d = 'potato'
e = foo(b,c,b)*d
f = spam(foo(a,b,e),2)
```

a. Draw the state of memory just before foo’s return command is executed for the second time. Include the global and any local frames. Marks will be deducted for any extra variables/frames.

Global Frame

- a → 2
- b → 1
- c → 0
- d → "potato"
- e → "potato"

foo() local frame

- a → 2
- b → 1
- c → 1

b. What will the value of the variable f be if the code is run to completion? If an error is generated, explain where and why it occurred.

The value of f will be 8 (integer). No error is generated.
Question 6
Consider the following Python program

```python
def foo(s):
    a = set()
    for x in s:
        if s.count(x) > 1:
            a.add(x)
    return a

a = 'snuffaluffagus'
b = set(a)
c = foo(a)
```

a. Draw the state of memory just before the call to `foo()` returns. Include the global and any local frames. Marks will be deducted for any extra variables/frames.

```
Global Frame

a → 'snuffaluffagus'
b → {'u', 'l', 'a', 'f', 'n', 'g', 's'}
```

```
Global Frame

s → 'snuffaluffagus'
a → {'u', 'f', 's', 'a'}
x → 's'
```

b. Draw the state of memory after the program finishes executing. Include the global and any local frames. Marks will be deducted for any extra variables/frames.

```
Global Frame

a → 'snuffaluffagus'
b → {'u', 'l', 'a', 'f', 'n', 'g', 's'}
c → {'u', 'f', 's', 'a'}
```

Question 7

Consider the following Python program

```python
a = {}
for n in pop:
    m = pop[n]
a[m] = n
b = max(a)
c = a[b]
```

a. Trace the above code and determine the values of a, b and c at the end. Marks will be deducted for any extra variables/frames.

Global Frame


b → 400
c → 'Beijing'
Question 8
Consider the following Python function. The input argument, `results`, is a dictionary with strings as keys and floats as values.

```python
def redline(results):
    times = []
    for racer in results:
        if results[racer] != 'Did not finish':
            times.append(results[racer])
        else:
            continue
    (winning_time, worst_time) = (min(times), max(times))
    times.remove(winning_time)
    second = min(times)
    if second - winning_time <= 5:
        print ('It was a close one!')
    return (winning_time, worst_time)
```

a. Summarize the function above.

This function takes a dictionary of racer names as keys and their race times as values. It returns a tuple with the best time as its first element and the worst time as its second. It prints “It was a close one!” if the first and second place racers were less than 5 minutes apart.

b. What would be the output of calling the command:

```python
print(redline({'JP': 32.3, 'Sonoshee': 39.4, 'Shinikai': 34.4, 'Frisbee': 'Did not finish'}))
```

(Assume the function has been defined first.)

It was a close one!
(32.3, 39.4)
**Question 9**

Briefly **summarize** what the following Python function does when called with a list of strings as its first argument and a small integer as its second argument.

```python
def foo(x, y):
    z = []
    for w in x:
        if len(w) >= y:
            z.append(w)
        else:
            z.append(w + ' ' * (y - len(w)))
    return z
```

The function returns a new list containing the strings in the original list in the same order, but with the following changes:

- Every string of the original list whose length is smaller than the second argument, the string is padded on the right with blanks to make its length equal to the second argument.

**Question 10**

Briefly **summarize** what the following Python function does when called with a dictionary as its argument.

```python
def grap(x):
    z = {}
    for y in x:
        z[x[y]] = y
    return z
```

It returns a new dictionary that is the reverse of its argument. That is, the keys of the new dictionary are the values of the old dictionary and each key m in the new dictionary is associated with a value n, if n is a key in the old dictionary and is associated with a value m in it.

Note that if a value m in the old dictionary is associated with many keys n1, n2, ..., nk, m will be associated with only one of them in the new dictionary.
Question 11
Consider the following Python code.

```python
def bar(string1, string2):
    result = []
    for ch in string1:
        if ch in string2:
            result.append(ch)
    return result

def foo(text, alpha):
    result = {}
    text = bar(text, alpha)
    for ch in text:
        result[ch] = result.get(ch, 0) + 1
    return result

text = "WELL, a += 20 is the same as a = a + 20 !"
r1 = foo(text, "aeiouy")
text = text.lower()
r2 = foo(text, "aeiouy")
```

a. Briefly summarize what the functions `bar` and `foo` do. (Note: This particular program takes many iterations to walk through step by step (it's 243 steps, actually!) – if you find it difficult to understand, try tracing through the function with shorter strings to start)

Function `bar` returns a list with the characters of its first argument that also appear in its second argument. Characters appear in the result in the same order and multiplicity that appear in the first argument.

Function `foo` returns a dictionary whose keys are the characters that appear in the function's first and second argument. The value associated with each character is the number of times this character appears in the first argument.

b) What are the values of `r1` and `r2` after this code is executed? (Note: Because this function takes so many iteration to walk through step by step, this is a good opportunity to practice predicting what the output will be for each function based on the inputs and the function description.).

```
r1 → {'a': 5, 'i': 1, 'e': 2}
r2 → {'a': 5, 'i': 1, 'e': 3}
```
Question 12
Consider the following Python code:

```python
def fun(vals):
    size = len(vals)
    if size == 0 :
        return None

    newvals= []
    for val in vals:
        newvals.append(val*val)

    newvals.sort()

    i = size // 2
    if size % 2 ==0:
        return (newvals[i-1] + newvals[i]) / 2.0
    else:
        return newvals[i]
```

a. Summarize what the following Python function does when it is called with a list of numeric values:

fun calculates and returns the median (middle value) of the squares of the values contained in its input list. If the input list is empty, fun returns None.

b) What would the call fun([ 5, -5, 2, -3, -10 ]) return?

25
Question 13

Use the function design recipe to create a Python function `filter()` that accepts as arguments two strings and returns a list that contains all the characters in its first argument that are not in the second argument. The characters in the list will appear in the same order and multiplicity (i.e. the same number of characters) as they appear in the first argument.

As a reminder, the function design recipe involves the following steps:
1. Examples.
2. Type contract
3. Header.
4. Description.
5. Body.
6. Test. (Using your examples)

```python
def filter(string1, string2):
    '''(string, string) -> list
    Accepts as arguments two strings and returns a list that contains all the characters in its first argument that are not in the second argument. The characters in the list will appear in the same order and multiplicity (i.e. the same number of characters) as they appear in the first argument.

    >>> filter('this', 'hi')
    ['t', 's']
    >>> filter('this', 'this')
    []
    >>> filter('a cat', 'dogs')
    ['a', ' ', 'c', 'a', 't']
    '''
    result = []
    for ch in string1:
        if ch not in string2:
            result.append(ch)
    return result

if __name__ == '__main__':
    print("result of filter('this', 'hi') should be ['t', 's']")
    print(filter('this', 'hi'))
    print("result of filter('this', 'this') should be []")
    print(filter('this', 'this'))
    print("result of filter('a cat', 'dogs') should be ['a', ' ', 'c', 'a', 't']")
    print(filter('a cat', 'dogs'))
```

Question 14
Use the function design recipe to create a Python function `percentages()` that accepts as arguments two strings and returns a dictionary whose keys are the characters that appear in the function's first argument but not in its second argument. Each character in the dictionary is associated with the percentage that this character appears in the first argument (i.e. the value is the fraction of the number of times the character appears in the first argument over the length of the argument).

As a reminder, the function design recipe involves the following steps:

1. Examples. (Make sure your tests cover all the conditional cases).
2. Type contract
3. Header.
4. Description.
5. Body.
6. Test. (Using your examples)

```python
def percentages(string1, string2):
    '''(string, string) -> dictionary

    Accepts as arguments two strings and returns a dictionary whose keys are the characters that appear in the function's first argument but not in its second argument. Each character in the dictionary is associated with the percentage that this character appears in the first argument/
    >>> percentages('this', 'Hi John')
    {'t': 0.25, 's': 0.25}
    >>> percentages('this', 'this')
    {}
    >>> percentages('a cat', 'dogs')
    {'t': 0.2, 'c': 0.2, 'a': 0.4, ' ': 0.2}
    '''
    result = {}
    length = len(string1)
    for ch in string1:
        if ch not in string2:
            result[ch] = result.get(ch, 0) + 1
    for key in result:
        result[key] = result[key]/float(length)
    return result

if __name__ == '__main__':
    print("result of percentages('this', 'Hi John') should be {'t': 0.25, 's': 0.25}")
    print(percentages('this', 'Hi John'))
    print("result of percentages('this', 'this') should be {}")
```
print(percentages('this', 'this'))
print("result of percentages('a cat', 'dogs') should be {'t': 0.2, 'c': 0.2, 'a': 0.4, ' ': 0.2}")
print(percentages('a cat', 'dogs'))
Question 15

a) Write the code for a function `median` that sorts its input list and calculates and returns the median (middle value) of the values in the list. If the list is empty, `median` returns `None`. If the list has an even number of values `median` also replaces the two middle values of the list with the median value. (i.e. the median value will now occur twice, replacing each of the two middle values)

As a reminder, the function design recipe involves the following steps:

1. Examples. (Make sure your tests cover all the possible return cases).
2. Type contract
3. Header.
4. Description.
5. Body.
6. Test. (Using your examples)

```python
def median(vals):
    '''(list) -> number

    Takes an input list, sorts the list, and calculates and returns the median (middle value) of the values in the list.
    If the list is empty, median returns None. If the list has an even number of values median also replaces
    the two middle values of the list with the median value

    >>> median([1,2,5,6,3])
    3
    >>> median([1,3,2,5])
    2.5
    >>> median([])

    *Note – to test for None, you typically leave a blank space. But for clarity on an exam you could also just write ‘None’.
    '''
    size = len(vals)
    if size == 0:
        return None
    vals.sort()

    i = size // 2
    if size % 2 == 0:
        j = (vals[i-1] + vals[i]) / 2.0
        vals[i-1] = j
```
vals[i] = j
    return j
else:
    return vals[i]

#Main
import doctest
print(doctest.testmod())

b) Suppose we execute the following code:

    list = [ 5, -5, 2, -2, -10, 8 ]
    result = median(list)

What would the values of result and list be?

    result ➞ 0.0

    list ➞ [-10, -5, 0.0, 0.0, 5, 8]
Question 16
Recall that in Python RGB images are represented by a two dimensional array of pixels. Each pixel is a tuple of three numbers representing the amount for red, green and blue color in the pixel. The position of each pixel is identified by the pair (column, row) which contains the column and row number for the pixel. The position of the pixel at the top-left corner is (0,0). The size of an image is also represented by the pair (width, height) containing the image’s width and length.

Assume that the following commands are executed:

```python
from PIL import Image

def vf(image):
    newimage = Image.new("RGB", image.size)
    (w, h) = image.size
    maxc = w - 1
    for r in range(h):
        for c in range(w):
            pix = image.getpixel((maxc - c, r))
            newimage.putpixel((c, r), pix)
    return newimage

# Main
image = Image.open("dog.jpg")
image = vf(image)
image.save("newdog.jpg")
```

Are the images in `newdog.jpg` and `dog.jpg` different after the program completes? If you believe that the two images are different, describe their differences.

The two images are different. The new image is a vertical flip of the original image. That is, the left side of the original image is the right side of the new image and the right side of the original image is the left side of the new image. For instance, if `dog.jpg` is the image on the left, `newdog.jpg` would be the image on the right:
Question 17
Write a function hf that accepts an image as its input and returns a new image which is the horizontal flip of the original image. That is, the top of the original image is the bottom of the new image and the bottom of the original image is the top of the new image. You do not need to include the docstring.

```python
def hf(image):
    newimage = Image.new("RGB", image.size)

    (w, h) = image.size
    maxr = h-1

    for c in range(w):
        for r in range(h):
            pix = image.getpixel((c, maxr-r))
            newimage.putpixel((c, r), pix)

    return newimage
```
Question 18

Summarize what the following Python function does when called with an image object:

```python
def hm(image):
    (w, h) = image.size

    newsize = tuple([w, 2*h])
    newimage = Image.new("RGB", newsize)

    for c in range(w) :
        for r in range(h):
            pix = image.getpixel((c,r))
            newimage.putpixel((c, r), pix)
    maxh = h-1
    for c in range(w) :
        for r in range(h):
            pix = image.getpixel((c, maxh-r))
            newimage.putpixel((c, h+r), pix)

    return newimage
```

The function `hm` accepts an image as its input and returns a new image which has the same width and two times the height of the original image. The upper half of the new image contains the old image and the lower half has the mirror image of the upper half. The original image is not changed.

For instance if `hm` is called with the left image as its argument, it will return the image shown on the right:
Question 19

Define a function `vm` which accepts an image as its input and returns a new image which has the same height and two times the width of the original image. The left half of the new image contains the old image and the right half has the mirror image of the left half. The original image is not changed. You do not need to write the docstring.

```python
def vm(image):
    (w, h) = image.size

    newsize = tuple([2*w, h])
    newimage = Image.new("RGB", newsize)

    for c in range(w):
        for r in range(h):
            pix = image.getpixel((c, r))
            newimage.putpixel((c, r), pix)

    maxw = w-1
    for c in range(w):
        for r in range(h):
            pix = image.getpixel((maxw-c, r))
            newimage.putpixel((w+c, r), pix)

    return newimage
```
Question 20

To answer the following two questions, it may be useful to know the following:

- When the `sorted()` function is applied to a group of tuples, the tuples in the group are ordered according to their first component. The tuples with the same first component are ordered according to their second component, and so on.

Assume that the following commands are executed:

```python
def foo(dict):
    result = {}
    for it in dict:
        val = dict[it]
        if val not in result:
            result[val] = []
        result[val].append(it)
    return result

# Main
students = { 'Susan': 90, 'John': 65, 'Mary': 90, 'Jeff': 50, 'Jane': 65, 'Lora': 95 }

students1 = foo(students)
students2 = sorted(students1.items())
```

What are the values of students1 and students2 when these commands complete?

- students1 → {65: ['Jane', 'John'], 50: ['Jeff'], 90: ['Susan', 'Mary'], 95: ['Lora']}

Note: Remember that the pairs in Student1 may be in different order as the dictionary does not maintain any order. This also means the lists of constructed names may also be in different orders, as the order will depend on the order the keys are accessed -- however, the order of names within the lists should be consistent between student 1 and student 2, as sorted() will sort by the first component, and will not sort the contents of each list.

- students2 → [(50, ['Jeff']), (65, ['Jane', 'John']), (90, ['Susan', 'Mary']), (95, ['Lora'])]
Question 21

Assume that the following commands are executed:

```python
def foo(dict):
    result = {}
    for it in dict:
        val = dict[it]
        if val not in result:
            result[val] = []
        result[val].append(it)
    return result

def bar(dict):
    x = sorted(dict.items())
    result = []
    for i in range(len(x)):
        key, vals = x[i][0], x[i][1]
        vals.sort()
        for val in vals:
            result.append(tuple([key, val]))
    return result

# Main
students = { 'Susan': 90, 'John': 65, 'Mary': 90, 'Jeff': 50, 'Jane': 65, 'Lora': 95 }

students1 = foo(students)
students2 = bar(students1)
```

What are the values of `students1` and `students2` when these commands complete?

- `students1` → `{65: ['Jane', 'John'], 50: ['Jeff'], 90: ['Mary', 'Susan'], 95: ['Lora']}

  Note: The pairs in Student1 may be in different order.

- `students2` → `[(50, 'Jeff'), (65, 'Jane'), (65, 'John'), (90, 'Mary'), (90, 'Susan'), (95, 'Lora')]`
Question 22

Write a function `wordFreq` that accepts as its argument the name of a text file and returns a dictionary whose keys are the words in the given file and its values are the frequencies for each word, i.e. the number of times the word appeared in the text file.

For instance, if you store the previous paragraph into a file and run `wordFreq` on it, it will return the dictionary:

```python
{'for': 1, 'word,' : 1, 'that': 1, 'as': 2, 'name': 1, 'appeared': 1, 'in': 2, 'the': 7, 'whose': 1, 'values': 1, 'a': 3, 'wordFreq': 1, 'keys': 1, 'of': 2, 'text': 2, 'each': 1, 'its': 2, 'times': 1, 'file': 2, 'given': 1, 'i.e.': 1, 'word': 1, 'file.': 1, 'accepts': 1, 'frequencies': 1, 'and': 1, 'dictionary': 1, 'argument': 1, 'number': 1, 'returns': 1, 'words': 1, 'Write': 1, 'function': 1}
```

You should design your function according to the recipe, but you don’t need to include examples / testing.

```python
def wordFreq(fileName):
    '''(string) -> dictionary
    Takes a filename and returns a dictionary whose keys are the words in the given file and its values are the frequencies for each word.
    '''
    file = open(fileName, "r")
    dict = {}

    for line in file :
        words = line.split()
        for word in words :
            if word in dict:
                dict[word] += 1
            else:
                dict[word] = 1
    return dict
```
Question 23
Consider a function, `getDict` that accepts as its argument the name of a CSV file that stores a dictionary and does the following: It opens the file, reads in the lines, creates the dictionary that was stored in the given file and returns that dictionary. The dictionary is stored in the file in the following manner: The first line of the file has the title "Keys, Values". Every other line contains a key and a value separated by a comma. Note that keys and values are both strings.

For instance if the file `opps.csv` contains the lines:

```
Key,Value
yes,no
front,back
back,front
true,false
right,left
right,wrong
```

The call `getDict("opps.csv")` will return:

```
{'yes': 'no', 'right': 'wrong', 'back': 'front', 'true': 'false', 'front': 'back'}
```

a. Explain why the 6th line in the example file (right, left) does not appear in the dictionary.

Dictionary keys are unique. The {right: left} entry is made when that line is processed, but the value ‘left’ overwritten the next time a pair with the key ‘right’ is encountered on line 7.

b. Implement the function described above. You should design you function according to the recipe, but you don’t need to include examples / testing.

```python
def getDict(fileName):
    file = open(fileName, "r")
    dict = {}

    # Skip first line
    file.readline()

    for line in file :
        line = line.strip()
        words = line.split("","
        dict[words[0]] = words[1]

    file.close()
    return dict
```
Question 24
Write a function `getDNAfreqs` that accepts as its argument the name of a FASTA file that stores a single DNA sequence and does the following: It opens the file, reads in the sequence, and creates and returns a dictionary with the frequencies of the DNA nucleotides A, C, G, T that appear in the sequence. The dictionary has another entry with the key “other” whose value is the number of characters other than A, C, G, T that appear in the sequence.

You should design your function according to the recipe, but you don’t need to include examples / testing. You should also use Biopython’s seq and SeqIO, but you may ignore the import part of the program. Just write the function.

```python
def getDNAfreqs(fileName):
    record = SeqIO.read(fileName, "fasta")
    dict = {}
    dict["other"] = 0

    for char in str(record.seq):
        if char in "ACGT":
            if char not in dict:
                dict[char] = 0
            dict[char] += 1
        else:
            dict["other"] += 1

    return dict
```
Question 25

Below are the contents of a sample file, list_for_breakfast.txt

> This is a grocery list for breakfast. You should ignore this line.
cheese
bacon
ham
eggs

Write the Python code for the body of the following function to behave according to what is described in its docstring. Assume that all input grocery lists follow the same format as the one shown above.

def checklist(current_items, grocery_list_file, verbose = False):

    """(list of strings, name of file, Bool) -> int

    Given a list of items which you already have in your grocery cart and a file containing the grocery items you were supposed to purchase, this function returns an integer number of the grocery list items that you've collected so far. If verbose is True, the function will also print a list containing the items in the list that still need to be collected. Items in your grocery cart which are not in your list will be ignored.

    >>> checklist(['cheese', 'bacon', 'ice' ], 'list_for_breakfast.txt')
    2

    >>> checklist(['cheese', 'bacon', 'ice' ], 'list_for_breakfast.txt',
              verbose = True)
    ['ham', 'eggs']
    2

    >>> checklist(['potatoes', 'bacon'] ,'list_for_breakfast.txt')
    1
    """
    remaining = []
    item_count = 0
    with open(grocery_list_file, 'r') as fh:
        for item in fh:
            if item.startswith('> '):
                pass
            else:
                if item.strip() in current_items:
                    item_count += 1
                else:
                    remaining.append(item.strip())
    if verbose == True:
        print(remaining)
    return item_count
**Question 26**

Consider the python code below:

```python
def bar(string_in):
    x = 0
    y = string_in.find('ata')

    while y != -1:
        x += 1
        y = string_in.find('ata', y+1)
    return x

def foo(list_in):
    a = []
    for c in list_in:
        if type(c) is list:
            # If c is a list.
            a.extend(foo(c))
        elif type(c) is str:
            # If c is a string.
            if bar(c) > 0:
                a.append(c)
            else:
                # If c is neither a list nor a string, enforce the
                # precondition by generating an error.
                assert False
    return a

p = ['tata', ['rsta', 'atat'], 'gatag']
q = p
p = foo(q)
```

a. What are the values of the variables `a` and `c` (in the frame of `foo()`) just before the `return` statement in `foo()` is executed for the last time? What are the values of the variables `p` and `q` (in the global frame) just after the last line completes.

```python
a = ['tata', 'atat', 'gatag']
c = 'gatag'
p = ['tata', 'atat', 'gatag']
q = ['tata', ['rsta', 'atat'], 'gatag']
```

b. Is `p` an alias for `q`?

*No: it cannot be an alias because `p` and `q` have different values. (Note that even if they had the same value, they still might not be aliases.)*
APENDIX
Some Useful Functions

Numbers
- \textbf{int}(x) : converts \texttt{x} to an integer
- \textbf{float}(x) : converts \texttt{x} to a floating point number

Strings, Lists and Tuples
- \texttt{x + y} : concatenates \texttt{x} and \texttt{y}
- \textbf{len}(x) : returns the length of \texttt{x}
- \texttt{s[i]} : returns the \texttt{i}-th item of \texttt{s}; \texttt{i} can be from 0 to \texttt{len(s)}-1
- \texttt{s[i:j]} : return the sub-part of \texttt{s} that starts from position \texttt{i} and ends at position \texttt{j}-1
- \texttt{s[i:]} : return the sub-part of \texttt{s} from position \texttt{i} to the end
- \texttt{s[:i]} : return the sub-part of \texttt{s} from the beginning to position \texttt{i}-1

Strings (only)
- \textbf{str}(x) : converts \texttt{x} to a string
- \texttt{s1 in s2} : returns True if string \texttt{s1} occurs as a substring in \texttt{s2}; returns False otherwise
- \textbf{s.count}(s1) : returns the number of non-overlapping occurrences of \texttt{s1} in \texttt{s}.
- \textbf{s.strip}() : returns the string \texttt{s} with any leading or trailing whitespace removed.
- \textbf{s.split}(sep) : returns a list of substrings constructed by splitting \texttt{s} at the separator character(s) given by string \texttt{sep}.
- \textbf{s.find}(s1, beg = 0) : returns the index of the first occurrence of substring \texttt{s1} in \texttt{s} after index \texttt{beg}. Returns \texttt{-1} if \texttt{s1} does not occur.

Lists (only)
- \texttt{x =[]} : creates an empty list
- \textbf{list}(collection) : creates an list from the elements of the collection (a list, set, or tuple)
- \texttt{list[i] = x} : sets the \texttt{i}-th item of \texttt{list} to the value \texttt{x}
- \textbf{list.append}(x) : adds \texttt{x} at the end of the list
- \textbf{list.extend}(collection) : extends list by appending each of the elements within collection.
- \textbf{list.insert}(i, item) : inserts item at position \texttt{i}
- `list.remove(item)`: removes item from the list
- `list.pop()`: removes the last item from the list
- `list.reverse()`: reverses the order of the items in the list
- `list.sort()`: sorts the items in the list

**Sets**
- `set()`: creates an empty set
- `set(collection)`: creates an set from the elements of the collection (a list, set, or tuple)
- `set.add(item)`: adds item to set
- `set.remove(item)`: removes item from set
- `set.clear()`: empties the set
- `set.intersection(o)`: returns a new set with the items which occur in both s and o.
- `set.union(o)`: returns a new set with the items which occur in either s and o.
- `set.difference(o)`: returns a new set with the items which occur in s but not in o.
- `set.issubset(o)`: returns True if and only if o contains all items in s.
- `set.issuperset(o)`: returns True if and only if s contains all items in o.

**Dictionaries**
- `x ={}`: creates an empty dictionary
- `dict[key]`: returns the value associated with this key
- `dict.get(key, default)`: returns the value for key if key is in the dictionary, else default. If default is not given, it defaults to None
- `dict.clear()`: empties the dictionary
- `dict.keys()`: returns a list with the dictionary keys
- `dict.values()`: returns a list with the dictionary values
- `dict.items()`: returns a list of (key, value) pairs with all the dictionary items

**Files**
- `file = open(file_name, mode)`: opens the file with the given file name;
- mode is "r" for reading, "w" for writing and "a" for appending
- when the file is opened for reading, `file` contains all the lines of the original file
  - `file.close()` : closes the file.
  - `file.readline()` : reads one entire line from the file.
  - `file.write(s)` : writes the string s to the file.

**Images**
- `Image.new("RGB", (width, height))` : creates a new blank image
- `Image.open("file_name")` : returns the image that is stored in the given file
- `image.save("file_name")` : saves image in the given file
- `image.size` : returns the size of the image, i.e. a tuple (w, h)
- `image.getpixel((x,y))` : returns the color for the pixel at position (x,y);
  the color is a triplet (r,g,b); r, g, b are from 0 to 255
- `image.putpixel((x,y), color)` : sets the color of pixel (x,y) to the given color
- `image.copy()` : creates a new copy of the image
- `image.resize(size)` : returns a new copy of the image resized to the given size. Size should be supplied as a tuple (x,y).

**Biopython’s Seq**
- all string operations apply
  - `seq.complement()` : returns the complement of seq
  - `seq.translate()` : returns the translation of seq
  - `seq.transcribe()` : returns the transcription of seq

**Biopython’s SeqRecord**
- components you can access: `seq, id, name, description`

**Biopython’s SeqIO**
- `SeqIO.read(handle, type)` : reads a SeqRecord from the handle
  - handle can be something that contains a sequence record, or can be the name of a file with a sequence record (type in this section will always be "fasta")
- SeqIO.parse(handle, type): reads a collection of SeqRecord’s from the file
  - handle can be something that contains a collection of sequence record, or can be the name of
    a file with a collection of sequence records
  - it returns the collection of the sequence records that were read
  - to access these records you can use the method next() or a for loop
- SeqIO.write(record, file, type): writes a SeqRecord to a file;
  - file must be opened before the write
- Entrez.efetch(db=db_name, rettype=type, id=sequence_ids)
  - Will return a handle to a collection of sequence records for the specified sequences

General
- input(string = "") : displays the string, reads a single line of text from the keyboard
  and returns it as a string.
- range(stop): Returns a sequence of integers from 0 to stop-1.
- range(start, stop): Returns a sequence of integers from start to stop-1.
- print(value1, value2, ..., sep = ' ', end = '
') : displays one or
  more values to the screen, separated by the character sep and finishing with the character
  end.
APPENDIX
Some Useful Syntax

Comparison Operators:  ==, !=, <=, <, >=, >

Boolean Operators:  and, or, not

Control Statements:

```
def function_name(inputs):
    block

for var in sequence:
    block

while condition:
    block

assert condition
```