Extra Practice Questions – Post Midterm Material

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></td>
</tr>
<tr>
<td>&gt;&gt;&gt; x.append('simon')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;&gt; x = [1, (23,)]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;&gt; x.extend('simon')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;&gt; x = [1, (23)]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;&gt; x + [99]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

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

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.
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.
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.

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.
**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
```

```python
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.

**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.
**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.
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.

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.)
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
```
**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
```
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)
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:

b) What would the call `fun([ 5, -5, 2, -3, -10 ])` return?
**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)
**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)
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)
b) Suppose we execute the following code:

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

What would the values of result and list be?
**Question 16**

Recall that in Python RGB images are represented by 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
**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.
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
```
**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.
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?
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?
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:

```
{'for': 1, 'word,' : 1, 'that': 1, 'as': 2, 'are': 1, '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': 2, '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.
**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.

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

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

Some Useful Functions

**Numbers**
- `int(x)` : converts x to an integer
- `float(x)` : converts x to a floating point number

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

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

**Lists (only)**
- `x = []` : creates an empty list
- `list(collection)` : creates an list from the elements of the collection (a list, set, or tuple)
- `list[i] = x` : sets the i-th item of list to the value x
- `list.append(x)` : adds x at the end of the list
- `list.extend(collection)` : extends list by appending each of the elements within collection.
- `list.insert(i, item)` : inserts item at position 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 = '\n')**: 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:

```python
def function_name(inputs):
    block

for var in sequence:
    block

while condition:
    block

assert condition
```

```python
if condition:
    block1
elif condition:
    block2
else:
    blockN
```