Compsci 101 Fall 2015 Exam 1 Rubric

Problem 1 (24 points)

Each answer is one point each. Spelling of int, integer doesn't matter. Similarly string, str doesn't matter. For the value column all answers must match exactly for one point EXCEPT, quotes don't matter for strings, e.g., "orange" and 'orange' are the same. Notice the opportunity for one point "bonus" for the comma shown in ['two', 'three'], but cannot receive more than 24 pts total. This point was not given if this particular entry was wrong.

fruits = ["apple", "pear", "orange", "banana", "blueberry"]
s = "one, two, three o'clock, four o'clock rock"

<table>
<thead>
<tr>
<th>a = s[5]</th>
<th>string</th>
<th>'t'</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = len(fruits)</td>
<td>int</td>
<td>5</td>
</tr>
<tr>
<td>z = s.count(&quot;&quot;,&quot;)</td>
<td>int</td>
<td>3</td>
</tr>
<tr>
<td>y = s.find(&quot;z&quot;)</td>
<td>int</td>
<td>-1</td>
</tr>
<tr>
<td>w = fruits[2]</td>
<td>string</td>
<td>'orange'</td>
</tr>
<tr>
<td>d = len(fruits) &gt; len(s)</td>
<td>boolean</td>
<td>False</td>
</tr>
<tr>
<td>e = s.split()[1:3]</td>
<td>list</td>
<td>['two', 'three']</td>
</tr>
<tr>
<td>g = 30/7</td>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>h = 27 % 12</td>
<td>int</td>
<td>3</td>
</tr>
<tr>
<td>q = 100 * 1.0 / 4</td>
<td>float</td>
<td>25.0</td>
</tr>
<tr>
<td>t = fruits[0][-1]</td>
<td>string</td>
<td>e</td>
</tr>
<tr>
<td>u = int(&quot;123&quot;)</td>
<td>int</td>
<td>123</td>
</tr>
<tr>
<td>v = fruits[-1]</td>
<td>string</td>
<td>'blueberry'</td>
</tr>
</tbody>
</table>
Problem 2 (8 points)

```python
def volume(side, height):
    return (math.sqrt(3) / (12)) * side * side * height
```

ALTERNATE SOLN:

```python
def volume(side, height):
    x = 1.73205 /12
    return x * side * side * height
```

Rubric:

+1 point for using sqrt(3) or 1.73205 in numerator and 12 in denominator of an expression
+2 points for multiplying side*side * height, note can be side**2 * height
+1 point for a return statement with a float that has some expression with side or height
+4 for return statement that uses side, height, and an expression with a numerator and denominator that is correct, either directly or with variables, but syntax errors ok, see below

You cannot lose points for something you don't get credit for!

-1 for a Python error that interferes with correctness,
no errors if code doesn't interfere with correctness or no points awarded
Example, -1 if used b and h instead of side and height
Problem 2 Part 2 (12 points)

```python
def zapfIndex(word):
    if len(word) < 4:
        return 2
    if word[-1] == 's':
        if word[-3:] == 'ies':
            return 10
        else:
            return 7
    if len(word) > 10 and word[-1] != 's':
        return 5
    return 13
```

Alternate Solution:

```python
def zapfIndex(word):
    if len(word) < 4:
        return 2
    if word[-3:] == 'ies':
        return 7
    if word[-1] == 's':
        return 7
    if len(word) > 10:
        return 5
    return 13
```

Note, code above is not the only way to be correct, for example there can be many uses of else in sequential if statements that are ok, and the else before return 7 isn't required, for example. The and word[-1] != 's' is not necessary above, for example

+1 compares len(word) to 4 and returns 2
+1 compares len(word) to 10 and returns 5

+1 word ending in s checked correctly, e.g., [-1] or [word.length()-1], in the context of returning 10 or 7, though correct return value doesn't matter, just trying
+1 word ending in 'ies' done correctly, e.g., [-3:] or .endswith("ies"), in context of returning 10 or 7 somehow, correctness of return value doesn't matter, just trying

+2 fall through of return 13 is clear in code, this is an attempt point, actual logic could mean it doesn't work, but attempt is clearly visible
+2 does 's' and 'ies' correctly for all words except for less than 4 or greater than 10

+4 function is totally correct for all paths and words, will require careful tracing

Example, -4 if don’t use return accept at end and thus only return 13 or 7 every time.
Example, -1 if no return at all
Problem 3 (8 points)

```python
def average(numbers):
    total = 0
    count = 0
    for n in numbers:
        total += n
        count += 1
    return total*1.0/count
```

General idea is that there are two counters, loop over numbers, accumulate values

+2 see loop over the list numbers, with some expressions using either total or count (doesn't have to be both. The list syntax can be wrong, but clearly a loop over values of list with expression in the loop body. **These are "right idea" points**

+2 loop syntax correct and includes a variable after for that's referenced in loop body

+4 code is totally correct, except for syntax errors that the compiler would catch

-1 Python error that earned points in rubric above,
-1 another "real" syntax issue, can lose 2 points if 2 or more earned, can't go negative with respect to points earned.

Example, -1 if no return
Example, -2 if use len as part of range - range(len(numbers))
Problem 4 Part A (8 points)

```python
def noVowels(word):
    answer = ""
    for ch in word:
        if isVowel(ch):
            answer = answer + "*
        else:
            answer = answer + ch
    return answer
```

Note, it's ok to write `isVowel(ch) == True`, and it's ok to use `+=` for concatenation.

+2 points for initializing a string to the empty string, using it inside a loop over words. Must have initialization and use in loop body **OR for using string inside loop and returning it** -- must have use in loop and either init or return.

+1 for returning a string that's used in the loop body **OR for initialization** (see above), must have earned the +2 above for this. If they're just missing init or missing return they lose this, but get the +2 above.

+1 for calling isVowel with **variable used in for loop**, doesn't matter if return value not used, or if syntax messed up, this is an idea point for calling isVowel.

+1 points for adding "*" correctly to string in the context of calling isVowel **OR for adding ch** if isVowel is false. Doing one or other other gets +2

+3 code is correct, except perhaps for syntax errors that are minor

-1 Python syntax issue/usage error that earned points. No deductions for code that didn't earn points
-1 can be repeated if needed, but only up to -2 and only if code earned points
Problem 4 Part b (8 points)

```python
def newList(words):
    answer = []
    for word in words:
        if isVowel(word[0])
            answer.append(noVowels(word))
        else:
            answer.append(word)
    return answer
```

ALTERNATE SOLUTION:

```python
def newList(words):
    answer = []
    for word in words:
        modified = word
        if isVowel(word[0])
            modified = noVowels(word)
        answer.append(modified)
    return answer
```

- +1 for creating a list that's empty and using it inside the loop **OR for using a list** in the loop body and returning it -- must be clearly a list
- +1 returning or initializing when the other +1 point were earned

- +1 for calling noVowels with variable used in for loop and using the value returned by noVowels
- +1 for calling isVowel and checking first character of variable used in loop, attempt is clear

- +2 adding variable used in loop to a list in the context of getting at least one of the previous two points or having no else

- +2 code is totally correct, except for syntax/usage error

-1 Syntax/usage error, but must have earned a point
Problem 5 Part 1 (8 points)

```python
def totalBill(adjective, fare):
    return fare + tipAmount(adjective)/100 * fare
```

- +2 call tipAmount with adjective as a parameter and store/use the result in any expression that seems related to tips (using tipAmount and adjective points)
- +2 add fare to some expression that seems related to tips (using fare points)
- +1 realizes should divide by 100, tries that related to value returned by tipAmount
- +3 expression is correct except for small errors, e.g., float etc

-1 syntax error/Python error, but only in context of earning points
Problem 5 Part 2 (8 points)

A -- why is the if statement there?

- +1 If statement discussed with words that indicate "to see if value in list", "adjective in words", supply default value
- +1 if states that calling .index results in an error when value not in list (expect this to be very rare point)

B -- use/purpose of .index

- +1 find index or location of adjective in list words
- +1 use this value to do something with other list -- must mention other use, not just what .index does, but doesn't have to be completely on target

C -- both lists used together

- +1 hints that locations in one list of adjectives correspond to tip amounts in other lists, uses words that are kind of on track
- +1 crisp explanation that words[i] and percentages[i] are closely related

D -- adding new value

- +1 explain that values will be added to lists, anything close to this gets this "try" point
- +1 clear in where values added, front, back, middle is ok, but clearly added in same location

Cannot earn points for D except for modifying lists
Problem 6 Part A (8 points)

```python
    def category(filename, category):
        answer = [ ]
        f = open(filename)
        for st in f:
            data = st.strip().split(",")
            foodtype = data[0]
            if foodtype == category:
                answer.append(data[1])
        return answer
```

+2 uses file correctly, but broken down as *(can earn 1 or 2 here)*
- +1 opens the file using filename parameter and does something with a loop involving files
- +1 clear in code that loop over the file processes line-at-a-time

+2 uses data[0] (or equivalent with other variable name than data) and compares it to parameter category *(can only earn 2 or 0 here)*
- +2 uses data[1] and does something with it related to adding to list, must be clear that value in "middle" of line is added to list for this point *(can only earn 0 or 2 here)*

+2 everything correct
- This is where points for initializing list missing, or not returning list missing. This is an all or nothing correctness, we've already given points for knowing how to initialize lists and use them properly

**DOES NOT matter if f.close() is called**, unless it's called inside the loop, that is a -1 Python error
Problem 6 Part B (8 points)

```python
def biggest(filename):
    largest = 0
    answer = ""
    f = open(filename)
    for st in f:
        data = st.strip().split","
        num = float(data[2])
        if num > largest:
            largest = num
            answer = data[1]
    return answer
```

ALTERNATE SOLUTION:

```python
def biggest(filename):
    largest = []
    names =[]
    f = open(filename)
    for st in f:
        data = st.strip().split","
        num = float(data[2])
        largest.append(num)
        names.append(names[0]);
    bignum = max(largest);
    pos = largest.index(bignum)
    return names[pos]
```

+4 logic for finding largest is evident, broken down as follows

- +1 for initializing a value and comparing it with if inside the loop and resetting it, this can be string or calories. This is an idea/logic point, not a syntax point
- +1 for correctly tracking max calories, that means use > appropriately with value of loop and reset it. This is an idea/logic point, not a syntax point
- +2 for keeping track of both calories and the food that had the calories, this means **must update in loop inside the if statement and return the value** -- it's ok to be wrong with float and data[2] here, this is max logic

+2 accesses data[2] and tries to convert to float (must be part of line)
- +1 tries to convert to float, or realizes it's a string and does something with it
• +1 correctly uses float(val) to convert -- this is independent of logic with if and max, just converting a value from the list of values to a float

+2 everything correct, minor Python syntax errors don’t matter, this is +2 for correct after other points

-1 for Python errors that are minor, mostly for final +2 points, conceivably if needed in other point-earning categories.

Note: You have to convert to a number, the string “123” is greater than the string “86”, the number can be float or int.