Compsci 101: Test 1

Robert Duvall

February 19, 2014

Name: ____________________________________

NetID/Login: _______

Community Standard Acknowledgment (signature) ____________________________

This test has 11 pages, be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts only 75 minutes.

In writing code you do not need to worry about specifying the proper import statements. Assume that all libraries and packages we have discussed are imported in any code you write.

We will try to give you the benefit of the doubt regarding inexact syntax in your written code where the meaning is unambiguous. However, it is very important that you indent your code properly; we cannot assume a line was intended to be indented if it is not clear.
PROBLEM 1:  (16 points)

Each of the variables below has a type and a value. The type is one of: list, boolean, int, string, or float. As an example, the type and value are shown in the first row of the table below. Fill in the other type and value entries based on the variable/expression in the first column.

<table>
<thead>
<tr>
<th>variable/expression</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>len([5,3,1])</td>
<td>int or integer</td>
<td>3</td>
</tr>
<tr>
<td>30.0 / 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>len(range(10,11))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;cat&quot; == &quot;dog&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;abcdef&quot;.find(&quot;b&quot;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>min(&quot;zebra&quot;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[&quot;a&quot;, 2, &quot;c&quot;, 4][2:]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;snow&quot; * 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>abs(-2.9)</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>
PROBLEM 2:  \((10\ points)\)

Draw the picture resulting from the following turtle commands. Include an arrow to represent the final position and direction of the turtle.

```python
for k in range(4):
    for x in range(4):
        if 0 < x < 3:
            snap.pendown()
        else:
            snap.penup()
        snap.forward(50)
        snap.left(90)
snap.left(90)
```
PROBLEM 3: (14 points)

Part A (5 points)
The function `isOverlapping` checks whether two appointment times overlap given four int parameters representing the start and ending hour times for two appointments. For simplicity, assume appointments are given in military time (0-24) and start on the hour.

Consider the following code to solve this problem:

```python
def isOverlapping(start1, end1, start2, end2):
    if start1 > start2:
        s = start1
    else:
        s = start2
    if end1 < end2:
        e = end1
    else:
        e = end2
    return s < e
```

Show the results of calling this function for the following calls:

```python
isOverlapping(10, 12, 11, 13)
isOverlapping(10, 11, 11, 12)
isOverlapping(10, 12, 14, 15)
```

Does this code work as intended? If not, explain why not.
Part B (5 points)

Your friend claims to have written a function that replaces each value in a list with twice the preceding value (and the first value with 0). For example, if the list [1, 2, 3] is passed as an argument, the function is supposed to turn it into [0, 2, 4].

Here is your friend’s the code:

```python
def doublePreceding(values):
    if len(values) > 0:
        previous = values[0]
        values[0] = 0
        for idx in range(1, len(values)):
            values[idx] = 2 * previous
            previous = values[idx]
    return values
```

Explain what the bug in this function is, and how to fix it.
Part C (4 points)

Consider the following function:

```python
def removeNegatives(numList):
    for num in numList:
        if num < 0:
            numList.remove(num)
```

When `removeNegatives([1, 2, 3, -3, 6, -1, -3, 1])` is executed, the result is `[1, 2, 3, 6, -3, 1]`. The loop traverses the elements of the list, and when a negative value (like -3 at position 3) is reached, it is removed, shifting the subsequent values one position earlier in the list (so 6 moves into position 3). The loop then continues on to process the next item, skipping over the value that moved into the removed items position. If there are two negative numbers in a row (like -1 and -3), then the second one will not be removed.

Explain how to (or actually rewrite) the code to avoid this problem.
PROBLEM 4:  (20 points)

Part A (9 points)
Write a function `middleTwice` that returns the middle letters of a given string repeated twice. If the given string has an even length, choose the two characters on either side of the midpoint. If it has an odd length, choose the three middle characters. If it is a single character, choose that single character.

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>middleTwice(&quot;Candy&quot;)</code></td>
<td>&quot;andand&quot;</td>
</tr>
<tr>
<td><code>middleTwice(&quot;and&quot;)</code></td>
<td>&quot;andand&quot;</td>
</tr>
<tr>
<td><code>middleTwice(&quot;snow&quot;)</code></td>
<td>&quot;nomo&quot;</td>
</tr>
<tr>
<td><code>middleTwice(&quot;s&quot;)</code></td>
<td>&quot;ss&quot;</td>
</tr>
</tbody>
</table>

Complete the function `middleTwice` below:

```python
def middleTwice(s):
    """
    return the middle characters of the given string repeated twice
    """
```

Part B (2 points)
Consider that the specification of the function `middleTwice` is changed to require a list instead of a string.
Here is an example of the new function call and results:

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>middleTwice([1, 2, 3, 4, 5])</code></td>
<td>[2, 3, 4, 2, 3, 4]</td>
</tr>
</tbody>
</table>

Explain what, if anything, needs to be changed in your implementation (e.g., what in your code is specific to the string type and needs to be changed to apply to a list type).
Part C (9 points)

Write a function `userIDsFromNames` that returns a list of strings, representing each person's user ID, when given two lists of strings, representing each person's first and last name respectively. A user ID is constructed by taking the middle characters of each person's first and last name, doubling them and then concatenating them together.

For example, given the following lists:

```python
```

The call `userIDsFromNames(firstNames, lastNames)` should return the following list:

```python
["bebevava", "weweracrac", "usausadgdg", "BobBobPhdPhd", "CCJJ", "CJCJJrJr"]
```

Complete the function `userIDsFromNames` below. You should call `middleTwice` and assume it works correctly.

```python
def userIDsFromNames (firstNames, lastNames):
    """
    return a list of user ID strings constructed from the given
    parallel lists of strings
    """
```
The APT function `txMsg` abbreviates words based on the following strategy:

- If the word is composed only of vowels, it is written exactly as in the original message.
- If the word has at least one consonant, write only the consonants that do not have another consonant immediately before them. Do not write any vowels.

For this problem, we are only considering abbreviating a single word, not an entire message. As with the APT, you can assume two functions exist and work correctly, `isvowel` and `isconsonant`, that return True only if the single given letter (i.e., string of length 1) is a vowel or consonant respectively.

This table illustrates what the function is supposed to return for each given input.

<table>
<thead>
<tr>
<th>call</th>
<th>return value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>txMsg(&quot;message&quot;)</code></td>
<td>&quot;msg&quot;</td>
</tr>
<tr>
<td><code>txMsg(&quot;please&quot;)</code></td>
<td>&quot;ps&quot;</td>
</tr>
<tr>
<td><code>txMsg(&quot;oooo&quot;)</code></td>
<td>&quot;oooo&quot;</td>
</tr>
</tbody>
</table>

Consider the following solutions that are not correct. Each both passes and fails at least one of the examples shown above.

Complete the table below for each solution, saying whether the given solution passes or fails the referenced test case and, if it fails, what the actual return value is. After the table explain, in general terms, what is wrong with the solution.

```python
def txMsg(word):
    if isvowel(word):
        return word
    result = ""  # empty string
    for index in range(len(word)):
        if index == 0 and isconsonant(word[0]):
            result += word[0]
        elif index > 0 and
            isconsonant(word[index]) and
            isvowel(word[index-1]):
            result += word[index]
    return result
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Pass/Fail</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why it fails:
Complete the table below for each solution, saying whether the given solution passes or fails the referenced test case and, if it fails, what the actual return value is. After the table explain, in general terms, what is wrong with the solution.

<table>
<thead>
<tr>
<th>ID</th>
<th>Pass/Fail</th>
<th>Actual Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why it fails:

```python
def txMsg(word):
    result = ""  # empty string
    vowelCount = 0
    previous = "e"
    for index in range(len(word)):
        if isvowel(word[index]):
            vowelCount += 1
            if vowelCount == len(word):
                return word
        if index > 1:
            previous = word[index-1]
        if isconsonant(word[index]) and
           isvowel(previous):
            result += word[index]
    return result
```