PROBLEM 1:  *(Types and values? (24 points) (ESTIMATE: 12 minutes))*

Consider the following variables and their values for the table below.

```
words = ['red', 'blue', 'tree', 'cup', 'dish']
phrase = '6 cars 9 boats for rent'
```

List in the table the type of variable and its value after being assigned the expression.

<table>
<thead>
<tr>
<th>variable = expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a = 'sandwich'</td>
<td>string</td>
<td>'sandwich'</td>
</tr>
<tr>
<td>b = a[4] + a[-2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c = words[-2] + words[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d = phrase[3] + phrase[-3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e = len(words)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f = len(a)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g = len(a) &lt;= len('house')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h = a[1:4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i = a.split('n')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>j = phrase.split()[3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k = 'be'.join(['top','cat', 'go'])</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 12//5 + 3.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = (8 % 5) + 5/4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 2:  (What is output? (8 points) (ESTIMATE: 8 minutes))

PART A (2 pts)

lst1 = [5, 2, 3]
lst1.append(7)
print(lst1)

Output:

PART B (2 pts)

lst2 = [5, 2, 3]
lst2.append([6, 4])
print(lst2)

Output:

PART C (2 pts)

lst3 = [5, 2, 3]
lst3 = lst3 + lst3
print(lst3)

Output:

PART D (2 pts)

lst4 = [5, 6, 1]
lst5 = lst4
lst4[1] = [7, 3]
print(lst5)

Output:

3
PROBLEM 3:  (Short code segments (16 pts) (Estimate: 12 minutes))

For each of the following five problems, use only what is indicated to set result to a Python expression. Do not use any Python methods or string constants unless indicated.

Here is an example.
Use str with indexing and the concatenation of two items to set result to the string 'by'.

```python
str = 'bicycle'
result = str[0] + str[3]
```

Note this answer uses only str, indexing, and the concatenation of two items.

Here is an example of a WRONG answer: `result = 'b' + 'y'` This answer is wrong because it uses string constants and it did not use str.

PART A (3 pts)
Use lst with indexing and the concatenation of two items to set result to the string 'sc'.

```python
lst = ['blue', 'fish', 'car']
result =
```

PART B (3 pts)
Use str with splicing and concatenation of two items to set result to the string 'yank'.

```python
str = 'junkyard'
result =
```
PART C (3 pts)
Use lst with indexing and splicing to set result to the string ‘pi’

```python
lst = [['red', 'blue', 'pink'], ['car', 'bus']]
result =
```

PART D (3 pts)
Use lst with indexing and the concatenation of two items to set result to the string ‘greentrain’

```python
lst = [['purple', 'red', 'green'], ['train', 'plane']]
result =
```

PART E (4 pts)
Use string str below with only join and split with appropriate small string constants to set result to ‘apple-yum-cherry-yum-plum’

You can use temporary variable(s) if you want to write the code in more than one line.

```python
str = ‘apple-*-cherry-*-plum’
result =
```
PROBLEM 4: *(Simple Function - Dinner Time: (6 points) (Estimate: 5 minutes))*

Write the function `dinnerPay` that has three float parameters: `cost` is the cost of the dinner, `taxrate` is the tax rate, and `discount` is a discount. The total price for dinner is the cost, plus the taxrate applied to the cost, minus the discount. You can assume the discount is always less than the cost. This function calculates and returns the total cost of the dinner, after applying the tax rate and the discount. Here are some examples:

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
</table>
| dinnerPay(35.00, .01, 10.00) | 25.35   | the tax is .01 times 35.00 = 0.35  
35.00 + 0.35 - 10.00 is 25.35 |
| dinnerPay(20.00, .05, 2.00)    | 19.0    | 20.00 + 1.00 - 2.00 is 19.0                |
| dinnerPay(60.00, .05, 10.00)  | 53.0    | 60.00 + 3.00 - 10.00 is 53.0               |

Complete the function `dinnerPay` below.

```python
def dinnerPay(cost, taxrate, discount):
```

```python
```
PROBLEM 5: \textit{(Simple Function - Compute word: (8 points) (Estimate: 7 minutes))}

Write the function \texttt{compute} that has two string parameters: \texttt{word1} and \texttt{word2}. This function returns \texttt{word1} with the digits ‘1’, ‘2’, and/or ‘3’ appended in that order at the end depending on the following.

- ‘1’ is added to the end of \texttt{word1} if \texttt{word1} is shorter in length than \texttt{word2}
- ‘2’ is added to the end of \texttt{word1} if \texttt{word1} comes before \texttt{word2} in alphabetical order
- ‘3’ is added to the end of \texttt{word1} if the first letter in \texttt{word2} is in \texttt{word1}

Here are some examples:

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute\texttt{('tiger', 'lion')}</td>
<td>‘tiger’</td>
<td>None of the three conditions are true</td>
</tr>
<tr>
<td>compute\texttt{('bear', 'eagle')}</td>
<td>‘bear123’</td>
<td>All three of the conditions are true</td>
</tr>
<tr>
<td>compute\texttt{('evolve', 'pokemon')}</td>
<td>evolve12</td>
<td>Only the first two conditions are true</td>
</tr>
</tbody>
</table>

Complete the function \texttt{compute} below.

\begin{verbatim}

def compute(word1, word2):

\end{verbatim}
PROBLEM 6 :  *(Random Random Food: (8 points) (Estimate: 8 minutes))*

Write the function \texttt{food} that has one parameter named \texttt{foodlist} that is a list of strings, with each string one word. This function calculates two values and then puts them together. First it randomly calculates one of the three strings ‘hot’, ‘warm’ or ‘cold’. Second it randomly chooses one of the strings from \texttt{foodlist} Then it returns a new string with the two calculated words in the order they were calculated, with a blank between them.

Here are some examples (there are other possible return values not shown):

<table>
<thead>
<tr>
<th>call</th>
<th>possible return</th>
<th>another possible return</th>
</tr>
</thead>
<tbody>
<tr>
<td>food(['egg', 'spinach'])</td>
<td>‘cold spinach’</td>
<td>‘hot egg’</td>
</tr>
<tr>
<td>food(['pasta', 'beans', 'bread'])</td>
<td>‘hot bread’</td>
<td>‘warm pasta’</td>
</tr>
<tr>
<td>food(['rice', 'potato', 'tea', 'soup'])</td>
<td>‘warm potato’</td>
<td>‘cold soup’</td>
</tr>
</tbody>
</table>

Complete the function \texttt{food} below.

```python
import random
def food(foodlist):
```

8
Consider the following function named `doubleup` that takes as input the string parameters `word` and `letter`, where `word` is one word and `letter` is a single letter. This function is supposed to return a copy of `word` with each occurrence of `letter` replaced by two occurrences of `letter`. This function does not work as intended.

```python
1 def doubleup(word, letter):
2     answer = ''
3     for ch in word:
4         if ch == letter:
5             answer = ch + ch
6         else:
7             answer = answer + ch
8     return answer
```

Here are some examples with the value returned and the correct value that should be returned.

<table>
<thead>
<tr>
<th>call</th>
<th>value returned</th>
<th>correct value</th>
<th>CORRECT?</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>doubleup('sister', 's')</code></td>
<td>'sster'</td>
<td>'ssisister'</td>
<td>WRONG</td>
</tr>
<tr>
<td><code>doubleup('earth', 'p')</code></td>
<td>'earth'</td>
<td>'earth'</td>
<td>CORRECT</td>
</tr>
<tr>
<td><code>doubleup('earth', 't')</code></td>
<td>'tth'</td>
<td>'earth'</td>
<td>WRONG</td>
</tr>
</tbody>
</table>

**PART A:** What would the code above return for the call `doubleup('sister', 't')`?

**PART B:** Give a call for the code `doubleup` above with a word and letter such that the letter is duplicated and the correct value is returned.

**PART C:** Explain how to correct the code by changing one line of code. Give the line number and the new line of code.
PROBLEM 8 :  (Loop the loop: (8 points) (Estimate: 7 minutes))

You must use a for loop to solve this problem. You cannot use the count function, or loop comprehension or while loop (topics we have not covered).

Write the function named `enough` that has three parameters: a string parameter named `word` that is one word, a string parameter named `letter` that is one letter, and an integer parameter named `count`. This function returns True if the letter occurs at least count times in `word`. Otherwise, this function returns False.

Here are several examples of calls to this function.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>enough('sisters', 's', 3)</code></td>
<td>True</td>
<td>'s' is in 'sisters' 3 times</td>
</tr>
<tr>
<td><code>enough('mississippi', 'i', 5)</code></td>
<td>False</td>
<td>'i' is in 4 times, not enough times</td>
</tr>
<tr>
<td><code>enough('videoconference', 'e', 3)</code></td>
<td>True</td>
<td>'e' is in 4 times, more than 3 times</td>
</tr>
</tbody>
</table>

Complete the function below.

```python
def enough(word, letter, count):
```

```python
def enough(word, letter, count):
```