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

```python
wlist = ['plate', 'yes', 'bug', 'card', 'no']
word = "keyboard"
phrase = "go to lunch now"
```

List in the table the type of variable and its value after being assigned the expression. The first one is done for you.

<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 = word[3] + word[-2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c = wlist[1] + wlist[3]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d = len(wlist)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e = len(word)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f = len(phrase) &lt; 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g = word[3:6]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h = wlist[1:2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i = phrase.split()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>j = phrase.split(&quot;n&quot;)[-1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k = &quot;or&quot;.join([&quot;g&quot;,&quot;cl&quot;,&quot;bat&quot;)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 18/5 + 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = (9 % 4) + 9/2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PROBLEM 2: \( \text{(What is output? (9 points) (ESTIMATE: 8 minutes))} \)

PART A (2 pts)

```python
lst1 = [9, 4]
lst1.append(4)
print(lst1)
```

Output:

PART B (2 pts)

```python
lst2 = [8, 1, 3]
lst2.append([2,7])
print(lst2)
```

Output:

PART C (2 pts)

```python
lst3 = [4, 8]
lst3 = lst3 + lst3
print(lst3)
```

Output:

PART D (3 pts)

```python
lst4 = [7, 4, 2]
lst5 = lst4
lst4[0] = [3]
lst5[-1] = 9
lst4 = lst4 + [8]
print(lst4)
print(lst5)
```

Output:

```
```
PROBLEM 3 :  \( \text{Short code segments (13 pts) (Estimate: 10 minutes)} \) 

For each of the following four 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 \text{str} with indexing and the concatenation of two items to set result to the string 'by'

\begin{verbatim}
str = 'bicycle'
result = str[0] + str[3]
\end{verbatim}

Note this answer uses only \text{str}, indexing, and the concatenation of two items.

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

\textbf{PART A (3 pts)}
Use \text{lst} with indexing and the concatenation of two items to set result to the string 'go'

\begin{verbatim}
lst = ['card', 'joy', 'sting']
result =
\end{verbatim}

\textbf{PART B (3 pts)}
Use \text{str} with splicing and concatenation of two items to set result to the string 'plate'.

\begin{verbatim}
str = 'duplicate'
result =
\end{verbatim}
PART C (3 pts)
Use lst with indexing and splicing to set result to the string ‘rip’

```python
lst = [['talk'], ['lantern', 'script'], ['cow']]
result =
```

PART D (4 pts)
Use string str below with only join and split with appropriate small string constants to set result to the value ‘Yo-go-Duke-go-Blue-go-Devils’

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

```python
str = "YogoDukegoBluegoDevils"
result =
```
PROBLEM 4: (What is my profit?: (6 points) (Estimate: 5 minutes))

The new online platform SellHere is a place you can sell items you make. Let \textbf{price} be the selling price of the item. The profit you make on the item is calculated in the following way. First you subtract a listing fee. Then SellHere gets a percentage of the remaining amount. You get the rest from SellHere. Your profit is the amount you get from SellHere minus your cost. It may be possible to have a negative profit.

Write the function \textbf{profit} that has four float parameters: \textbf{price} is the sell price of the item, \textbf{listfee} is the listing fee, \textbf{percent} is the percentage SellHere gets after subtracting the listing fee, and \textbf{cost} is the cost to make the item. This function returns the profit. You can assume arguments for the four parameters are greater than 0.0, and the parameter \texttt{perc} is less than or equal to 50.00 (50.00 would mean 50 percent).

Here is an example:

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>profit(30.00, 0.50, 15.0, 10.00)</td>
<td>15.075</td>
<td>remove listing fee from 30.00 results in 29.50, 15% of 29.50 is 4.425, resulting in 25.075 profit is 25.075 - 10.00 which is 15.075</td>
</tr>
</tbody>
</table>

Complete the function \textbf{profit} below.

\begin{verbatim}
def profit(price, listfee, percent, cost):
\end{verbatim}
For some game there are two teams, team\textsubscript{1} and team\textsubscript{2}, with team\textsubscript{1} receiving score\textsubscript{1} and team\textsubscript{2} receiving score\textsubscript{2}. Write the function \texttt{winner} that has four parameters: \texttt{team1} is a string for the name of team\textsubscript{1}, \texttt{score1} is an integer representing the score for team\textsubscript{1}, \texttt{team2} is a string for the name of team\textsubscript{2}, and \texttt{score2} is an integer representing the score for team\textsubscript{2}. This function returns the following string (see examples for the correct format): 1) if team\textsubscript{1} had more points return a string with team\textsubscript{1}'s name saying they won and by how much, 2) if team\textsubscript{2} had more points return a string with team\textsubscript{2}'s name saying they won and by how much, and 3) if it was a tie, return a string indicating the game was a tie and the number of points each team had.

Here are some examples:

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{winner(&quot;Duke&quot;, 63, &quot;UNC&quot;, 57)}</td>
<td>'Duke won by 6'</td>
</tr>
<tr>
<td>\texttt{winner(&quot;Virginia&quot;, 67, &quot;NCSU&quot;, 74)}</td>
<td>'NCSU won by 7'</td>
</tr>
<tr>
<td>\texttt{winner(&quot;Miami&quot;, 71, &quot;Clemson&quot;, 71)}</td>
<td>'Game ended in a tie. Both teams had 71'</td>
</tr>
</tbody>
</table>

Complete the function \texttt{winner} below.

\begin{verbatim}
def winner(team1, score1, team2, score2):
    # Your implementation goes here
\end{verbatim}
PROBLEM 6:  *(Random Word Combo: (8 points) (Estimate: 8 minutes))*

Write the function `pick` that has three string parameters named `word1`, `word2`, and `word3`. This function returns a string that is randomly one of three strings:

- a new word that is `word1` followed by `word2`
- a new word that is `word1` followed by `word3`
- a new word that is `word2` followed by `word3`

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>pick(&quot;car&quot;, &quot;boat&quot;, &quot;train&quot;)</td>
<td>‘carboat’</td>
<td>‘cartrain’</td>
</tr>
<tr>
<td>pick('blue','red','green')</td>
<td>‘redgreen’</td>
<td>‘bluered’</td>
</tr>
<tr>
<td>pick('tennis','golf','bowling')</td>
<td>‘tennisbowling’</td>
<td>‘golfbowling’</td>
</tr>
</tbody>
</table>

Complete the function `pick` below.

```python
import random

def pick(word1, word2, word3):
    # Your implementation here
```
PROBLEM 7: (What is happening with this code?: (8 points) (Estimate: 7 minutes))

Consider the following numbered code, and then answer the questions below.

```python
1 def course(name, time):
2     return "I take " + name + " at " + time
3
4 def meal(place, type):
5     return "I eat at " + place + " for " + type
6
7 def life(whoami, task, item1, item2):
8     me = "I am " + whoami
9     return me + "\n" + task + item1 + item2
10
11 if __name__ == '__main__':
12     print(meal("Enzos", "Dinner"))
13     print(course("Econ101", "8:30am"))
14     print(life("Johnson", meal, "Sazon", "lunch"))
15     print(life("Purnima", course, "CompSci101", "10:15am" ))
```

A. What is the output when line 12 prints?

B. What is the output when line 13 prints?

C. What are the types of the first two arguments in line 14?

D. There is an error in the program on line 9. The output for lines 14 and 15 should be:

I am Johnson
I eat at Sazon for lunch
I am Purnima
I take CompSci101 at 10:15am

What is the correct line 9 so the program works correctly?
PROBLEM 8:  (Building a new word: (8 points) (Estimate: 8 minutes))

You must use a for loop to solve this problem. Here is an example for loop that returns the sum of the numbers in a list, so `addlist([6, 2, 8])` returns 16.

```python
def addlist(somelist):
    sum = 0
    for num in somelist:
        sum = sum + num
    return sum
```

Write the function named `build` that has three string parameters: one named `word` that is one word, and two named `let1` and `let2` that are both a string of a single letter.

This function returns a new word that is `word` but with every occurrence of `let1` doubled if `let1` appears in `word`, and every occurrence of `let2` removed if `let2` appears in `word`. Note `let1` and `let2` will not be the same letter.

Here are several examples of calls to this function. Complete the function below.

<table>
<thead>
<tr>
<th>call</th>
<th>returns</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>build(&quot;fold&quot;,&quot;o&quot;,&quot;l&quot;)</code></td>
<td>'food'</td>
<td>&quot;o&quot; is doubled and &quot;l&quot; is gone</td>
</tr>
<tr>
<td><code>build(&quot;bottle&quot;,&quot;o&quot;,&quot;t&quot;)</code></td>
<td>'boole'</td>
<td>&quot;o&quot; is doubled and &quot;t&quot;'s are gone</td>
</tr>
<tr>
<td><code>build(&quot;mississippi&quot;, &quot;i&quot;, &quot;k&quot;)</code></td>
<td>'miissiissiippii'</td>
<td>all &quot;i&quot;'s doubled</td>
</tr>
</tbody>
</table>

```python
def build(word, let1, let2):
```
# Python Reference Sheet for Compsci 101, Exam 1, Spring 2023

You must turn in this sheet with your exam. Do not write anything to grade on this sheet.

## Mathematical Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
<td>$4 + 5 = 9$</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
<td>$9 - 5 = 4$</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
<td>$3 \times 5 = 15$</td>
</tr>
</tbody>
</table>
| / and // | division | $\frac{6}{3} = 2.0$
|        |          | $\frac{6}{4} = 1.5$
|        |          | $\frac{6}{4} = 1$ |
| %      | mod/remainder | $5 \% 3 = 2$ |
| **     | exponentiation | $3^{2} = 9$, $2^{3} = 8$ |

## String Operators

| +      | concatenation | "ab" + "cd" = "abcd" |
| *      | repeat | "xo" * 3 = "xoxoxo" |

## Comparison Operators

| ==     | is equal to | 3 == 3 is True |
| !=     | is not equal to | 3 != 3 is False |
| >=     | is greater than or equal to | 4 >= 3 is True |
| <=     | is less than or equal to | 4 <= 3 is False |
| >      | is strictly greater than | 4 > 3 is True |
| <      | is strictly less than | 3 < 3 is False |

## Boolean Operators

| x=5
not | flips/negates the value of a bool | (not x == 5) is False |
| and | returns True only if both parts of it are True | (x > 3 and x < 7) is True |
| or  | returns True if at least one part of it is True | (x < 3 or x > 7) is True |

## Type Conversion Functions

| int(x) | turn x into an integer value | int("123") == 123 |
|        | int can fail, e.g., int("abc") raises an error |
| float(x) | turn x into a float value | float("2.46") == 2.46 |
|         | float can fail, e.g., float("abc") raises an error |
| str(x)  | turn x into a string value | str(432) == "432" |
| type(x) | the type of x | type(1) == int |
|         | type(1.2) == float |

## Strings
### String Functions

**s="colorful"**

<table>
<thead>
<tr>
<th>Name</th>
<th>Returns</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>.split()</td>
<td>list of &quot;words&quot; in s</td>
<td>&quot;big bad dog&quot;.split() == [&quot;big&quot;, &quot;bad&quot;, &quot;dog&quot;]</td>
</tr>
<tr>
<td>.split(&quot;&quot;,&quot;&quot;)</td>
<td>list of &quot;items &quot; in s that are separated by a comma, e.g., s.split(&quot;;&quot;) will split on a colon and s.split(&quot;gat&quot;) will split on the string &quot;gat&quot;.</td>
<td>&quot;this,old,man&quot;.split(&quot;;&quot;) == [&quot;this&quot;, &quot;old&quot;, &quot;man&quot;]</td>
</tr>
<tr>
<td>''.join(lst)</td>
<td>concatenate elements of lst, a list of strings, separated by '' or any string</td>
<td></td>
</tr>
</tbody>
</table>

### Miscellaneous Functions

- **help(x)** documentation for module x
- **len(x)** length of sequence x, e.g., String or List
  - len("duke") == 4

### List Functions

- **lst = [3, 6, 8, 1, 7]**

<table>
<thead>
<tr>
<th>Name</th>
<th>Returns</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>lst[x]</td>
<td>index an element</td>
<td>lst[0] == 3</td>
</tr>
<tr>
<td>lst[-1]</td>
<td></td>
<td>lst[-1] == 7</td>
</tr>
<tr>
<td>lst[x:y]</td>
<td>splice of list, sublist from index x up to but not including index y</td>
<td>lst[1:3] == [6, 8]</td>
</tr>
<tr>
<td>+ operator</td>
<td>concatenate two lists</td>
<td>[3,4] + [1,3,2] == [3,4,1,3,2]</td>
</tr>
</tbody>
</table>

### Random Functions (import random)

- **random.choice(list_of_choices)** returns a random element from list_of_choices. Gives an error if list_of_choices has length 0.
- **random.randint(start, end)** Returns a random integer between start and end. Unlike range() and list slicing, the largest value it can return is end, not end-1.
- **random.random()** Returns a random float between 0 and 1.