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

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

```
groceries = ["butter", 3, "milk", 2, "eggs", 12, ["cheddar", "mozzarella", "feta"]]
```

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

variable = expression	Type	Value
`a = "sandwich"`	string	`"sandwich"`
`b = len(groceries)`		
`c = len(groceries[-1][-1]) # Negative indices`		
`d = groceries[0] + groceries[2]`		
`e = groceries[:6]`		
`f = "-".join(groceries[-1]) # Negative index`		
`g = "HelloWord".split("o")`		
`h = "Eggs" in groceries`		
`i = groceries[0] < groceries[2]`		
`j = groceries[6][0] * 2`		
`k = 9 // 2`		
`l = 8 / 4 * 2`		
`m = 3 + 6 % 2 ** 2`		

PROBLEM 2 : (*Short Code (16 pts) (Estimate: 8 minutes)*)

For problems A and B, craft a Python expression using only the specified instructions to set a value to result. For example:

> Use `phrase` with indexing, slicing, concatenation, join, and/or split to set `result` to the string `"by"`.
>
> ```
> phrase = "bicycle"
> result = phrase[0] + phrase[3]
> ```
>
> Notice how this answer does not simply assign the string `result = "by"`.

Part A (4 points)

Use `name` with indexing, slicing, concatenation, join, and/or split to set `result` to the string `"EdranShe"`.

```
name = "Ed Sheeran"

result =
```

Part B (4 points)

Use `word` and `songs` with indexing, slicing, concatenation, join, and/or split to set `result` to the string `"Perfect and Bad Habits and Shivers"`.

```
word = " and "
songs = ["Perfect", "Bad Habits", "Happier", "Shivers"]

result =
```

For problems C and D, write the output of the code.

Part C (4 points)

```
songs = ["Perfect", "Bad Habits", "Happier", "Shivers"]

x = songs

x[0] = "Thinking Out Loud"

print(songs)
```

Output:

Part D (4 points)

```
groceries = ["butter", 3, "milk", 2, "eggs", 12, ["cheddar", "mozzarella", "feta"]]

y = groceries[6]

y = y + ["gouda"]

print(groceries)
```

Output:

PROBLEM 3 : (***Spot the Bug (15 points) (Estimate: 15 minutes)***)

The following code is meant to evaluate temperature values and print a single message for each, indicating whether it's hot, warm, cool, or cold.
- Temperatures 10 and below are cold.
- Temperatures between 11 and 20 (inclusive of both) are cool.
- Temperatures between 21 and 30 (inclusive of both) are warm.
- Temperatures 31 and above are hot.

The problem is that the function contains bugs! Review the code below:

```
01   """
02   This program processes temperatures.
03   """
04
05   def check_temperature(temperatures):
06      for temp in temperatures:
07          if temp > 30:
08              print("It's hot!")
09          if temp > 20:
10              print("It's warm!")
11          if temp > 10:
12              print("It's cool!")
13          else:
14              print("It's cold!")
15
16   if __name__ == "__main__":
17      temps = [60, 20, 10]
18      check_temperature(temps)
19      print("All done!")
```

Assuming the code worked correctly, the expected output of this code is:

```
It's hot!
It's cool!
It's cold!
All done!
```

The temperature value of 60 is hot, 20 is cool, and 10 is cold. The actual output of the code differs due to bugs in the code. Answer the following questions regarding the above code.

a) What line number contains the function call?

b) What line number contains the function definition header?

c) What is the name of the argument?

d) What is the name of the parameter?

e) What is the value of the `for` loop variable `temp` in the buggy code when the first iteration of the loop finished executing?

f) What is the value of the `for` loop variable `temp` in the buggy code when the last iteration of the loop finished executing?

g) What value does this function return? If nothing is returned, write the value `None`.

h) What is the actual output (if any) when this buggy program is executed? If nothing is displayed, write `"nothing"`. If an error is displayed, name or describe the error. Recall that output is what is shown in the python console window.

Problem continues on next page.

i) Using the space below, rewrite the function with the error(s) removed.

PROBLEM 4 : (*Random Animal Classifier (12 points) (Estimate: 8 minutes)*)

Write a function named `classify` that takes one parameter, `animals`, which is a list of strings. This function should use a random number to randomly pick an animal from the list. It should then classify the randomly picked animal as a `"Pet"`, `"Wild"`, or a `"Farm"` animal. An animal can have multiple classifications. The return value of this function should be a list of strings, where the animal's classifications are the elements of the list.

The following animals should be classified as `Pet`: `cat`, `dog`, `rabbit`, `pigeon`.
The following animals should be classified as `Wild`: `lion`, `tiger`, `elephant`, `rabbit`, `pigeon`
Any other animal should be classified as `Farm`.

Notice how `rabbit` and `pigeon` are both `Pet` and `Wild`. All other animals have only one classification.

Here are the descriptions of potential return values for this function.

Function Call	**Random Animal**	**Return value**
`classify(["rabbit", "cat", "cow"])`	`"rabbit"`	`["Pet", "Wild"]`
`classify(["rabbit", "cat", "cow"])`	`"cow"`	`["Farm"]`
`classify(["tiger", "dog"])`	`"dog"`	`["Pet"]`
`classify(["tiger", "dog", "pigeon", "horse"]`	`"pigeon"`	`["Pet", "Wild"]`

Complete the function below.

```
import random

def classify(animals):
```

PROBLEM 5 : (*Step aside join function, I'm going solo (16 points) (Estimate: 10 minutes)*)

Let's write our own `join` function. Write the function named `combine_words` that takes two parameters: `lst_of_words` (a list of strings) and `word` (a string). This function should join all the strings in `lst_of_words` into a single string, inserting `word` in between each element in `lst_of_words`. **You are NOT allowed to use the join function, that would defeat the purpose of this question and would result in a 0 for this question.**

Here are several examples of calls to this function:

Function call	**Return value**	**Comment**
`combine_words(["hello", "world"], "#")`	`"hello#world"`	Notice the `#` is in between `hello` and `world`
`combine_words(["duke", "compsci", "101"], "!")`	`"duke!compsci!101"`	Notice the `!` is in between each element.
`combine_words(["python"], "$")`	`"python"`	Since there is only 1 element, the `$` is not used.
`combine_words([""], "*")`	`""`	An empty list should return an empty string.

Complete the function below.

```
def combine_words(lst_of_words, word):
```

PROBLEM 6 : (*Sum of Numbers (16 points) (Estimate: 12 minutes)*)

Write a function named `sum_of_nums` that takes two integer parameters: `start` and `end`. The function should return the sum of all numbers in the range from `start` to `end` (inclusive of both) that are evenly divisible by either `3` or `5`.

Here are several examples of calls to this function.

Function Call	**Return value**	**Explanation**
`sum_of_nums(1,5)`	8	The numbers 3 and 5 are divisible by 3 or 5, and their sum is 8.
`sum_of_nums(1,10)`	33	The numbers 3, 5, 6, 9, and 10 are divisible by 3 or 5, and their sum is 33.
`sum_of_nums(10,20)`	75	The numbers 10, 12, 15, and 18 are divisible by 3 or 5, and their sum is 75.

Complete the function below.

```
def sum_of_nums(start, end):
```