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

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

```
wlist = ['book', 'pencil', 'pen', 'ruler', ['blue', 'red', 'pink']]
word = "spiderbird"
phrase = "go duke win big"
```

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 = 'cat'`	string	`'cat'`
`b = word[2] + word[-4]`		
`c = wlist[2] + wlist[-2]`		
`d = len(wlist)`		
`e = wlist[4][0] * 2`		
`f = wlist[1] < wlist[2]`		
`g = word[:3] + word[-2:]`		
`h = wlist[1:3]`		
`i = phrase.split()`		
`j = word.split("i")[-1]`		
`k = "*".join(wlist[-1])`		
`m = 14//6 + 3.1`		
`n = (11 % 4) + 3/2 + 2**3`		

PROBLEM 2 : (***What is output? (9 points) (ESTIMATE: 8 minutes)***)

PART A (2 pts)

```
lst1 = [6, 4]
lst1.append(3)
print(lst1)
```

Output:

PART B (2 pts)

```
lst2 = [5, 1, 4]
lst2.append(["go"])
print(lst2)
```

Output:

PART C (2 pts)

```
lst3 = [5] + [8]
lst3 = lst3 + lst3
print(lst3)
```

Output:

PART D (3 pts)

```
lst4 = [6, 3, 12, 4]
lst5 = lst4
lst4[0] = 2
lst5[-1] = [7]
lst4 = lst4 + [10]
print(lst4)
print(lst5)
```

Output:

PROBLEM 3 : (***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 str with indexing and the concatenation of two items to set result to the string 'by'

```
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 = str[:1] + 'y'` This answer is wrong because it uses splicing and a string constant and it did not use indexing with str.

PART A (3 pts)

Use **lst** with indexing and the concatenation of two items to set result to the string 'be'

```
lst = ['leaf', 'bun', 'ghost']

result =
```

PART B (3 pts)

Use str with splicing and concatenation of two items to set result to the string 'chow'.

```
str = 'flowchart'

result =
```

PART C (3 pts)

Use lst with indexing and splicing to set result to the string 'ring'

```
lst = [['bell'],['spurring', 'window'], ['rhyme']]

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 `'taco-gazpacho-burrito-tamale'`

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

```
str = "tacogazpachoburritotamale"




result =
```

PROBLEM 4 : (*How many chairs?: (6 points) (Estimate: 5 minutes)*)

Write the function `chairs` that has three parameters: an integer **total** that is the number of chairs in a cafeteria, and two floats **perc1** and **perc2** that represent percentages (if perc1 is 0.15 that represents 15%). Some of the chairs in the cafeteria need to be removed, **perc1** percent of them are being removed for repairs, and **perc2** percent of them are being removed for an event in another room. This function returns the number of chairs remaining after **perc1** percent and **perc2** percent of the chairs have been removed. If the number to be removed is not an integer number then truncate it. That is, if 2.8 percent of the chairs need to be removed for repairs, that means 2 chairs need to be removed. You can assume all the arguments for the three parameters are greater than 0, and the two percentages total will not remove all the chairs.

Here are examples:

call	**returns**	**comment**
chairs(120, 0.10, 0.21)	83	remove 12 and 25 chairs, 120*0.10 is 12.0, 120* 0.21 is 25.2
chairs(45, 0.03, 0.06)	42	remove 1 and 2 chairs, 45 * 0.03 is 1.35, 45 * 0.06 is 2.7

Complete the function **chairs** below.

```
def chairs(total, perc1, perc2):
```

PROBLEM 5 : (***Player statistics: (8 points) (Estimate: 7 minutes)***)

Write the function `stats` that has four parameters: **opponent** is a string for the name of the opposing team for a Duke basketball game, **player** is a string for the name of a Duke player, **points** is an integer for the number of points the player made in that game, and **rebounds** is the number of rebounds the player got in that game. This function returns a string containing the name of the opponent, followed by a colon ":", followed by the name of the player, followed by a colon ":", followed by a message. If the player got a double-double (they got 10 or more points and 10 or more rebounds), the message is "double-double". Otherwise, if the player had 10 or more points the message is "10 or more points", if the player had 10 or more rebounds the message is "10 or more rebounds", and if none of those are true, then the message is "practice harder". Here are some examples:

call	**returns**
stats("Auburn", "James", 9, 5)	'Auburn:James:practice harder'
stats("Miami", "Maluach", 12,15)	'Miami:Maluach:double-double'
stats("Louisville", "Brown", 6, 11)	'Louisville:Brown:10 or more rebounds'

Complete the function **stats** below.

```
def stats(opponent, player, points, rebounds):
```

PROBLEM 6 : (*Random Word Combo: (8 points) (Estimate: 8 minutes)*)

Write the function `compose` that has four string parameters named **word1**, **word2**, **word3**, and **word4**.

This function returns a string that is randomly one of word1 or word2, followed by randomly one of word3 or word4.

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

call	**possible return**	**another possible return**
compose(”blue”, ”green”, ”lake”, ”river”)	‘blueriver’	‘greenlake’
compose(‘icy’,‘hot’,‘tea’,‘coffee’)	‘icycoffee’	‘icytea’

Complete the function **compose** below.

```
def compose(word1, word2, word3, word4):
```

PROBLEM 7 : (*What is happening with this code?: (8 points) (Estimate: 7 minutes)*)

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

```
1   def choose2(wordlist):
2       index = random.randint(1,len(wordlist))
3       newPhrase = wordlist[index]
4       index = random.randint(1, len(wordlist))
5       newPhrase = newPhrase + wordlist[index]
6       return newPhrase
7
8   if __name__ == '__main__':
9       list1 = ["little", "big", "huge", "tiny", "small"]
10      print(choose2(list1))
11      list1 = ["one", "four", "six"]
12      print(choose2(list1))
```

A. What is an example of output when line 10 prints?

B. What is an example of output when line 12 prints?

C. What is the type of the argument in line 10 for the **choose2** function?

D. Explain what this code does or is supposed to do.

E. Most of the time, the program runs fine with no errors. But sometimes the program crashes with this error: *IndexError: list index out of range*
Which line(s) of code might the error be from?

F. Explain what the error is.

G. Explain how to correct this error (the correct code and where it goes), so the program runs correctly. You should not rewrite the code. Make only small changes to fix it.

PROBLEM 8 : (***Long words: (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.

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

Write the function named **longer** that has two parameters: a list of strings named **wordlist**, and an integer named **limit**. This function returns the number of words in wordlist that are longer than limit. Here are several examples of calls to this function. Complete the function below.

call	**returns**	**comment**
longer([‘blue’, ‘lightgreen’, ‘pink’, ‘red’, ‘yellow’], 5)	2	2 words are length > 5
longer([‘blue’, ‘lightgreen’, ‘pink’, ‘red’, ‘yellow’], 10)	0	no words are length > 10
longer([‘cow’,‘horse’,‘chicken’,‘snake’,‘eel’,‘elephant’], 4)	4	4 words are length > 4

```
def longer(wordlist, limit):
```

Python Reference Sheet for Compsci 101, Exam 1, Spring 2025

MUST TURN IN! WE WILL NOT GRADE ANYTHING YOU WRITE ON THE Reference Sheet

Mathematical Operators		
Symbol	**Meaning**	**Example**
+	addition	4 + 5 = 9
-	subtraction	9 - 5 = 4
*	multiplication	3*5 = 15
/ and //	division	6/3 = 2.0 6/4 = 1.5 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 (x > 3 and x > 7) is False
or	returns True if at least one part of it is True	(x < 3 or x > 7) is False (x < 3 or x < 7) is True
Type Conversion Functions		
int(x)	turn x into an integer value	int("123") == 123 int(5.8) == 5
	int can fail, e.g., int("abc") raises an error	
float(x)	turn x into an 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

String Index and Splicing		
`s="colorful"`		
		Example
s[x]	index a character	s[0] == 'c' s[-3] == 'f' s[5] == 'f'
s[x:y]	splice of string, substring from index x up to but not including index y	s[2:5] == 'lor' s[:5] == 'color' s[4:-1] == 'rfu' s[5:] == 'ful'

String Functions		
`s="colorful"`		
Name	**Returns**	**Example**
.split()	list of "words" in s	"big bad dog".split() == ["big","bad", "dog"]
.split(",")	list of "items " in s that are separated by a comma *In general can split on any string, not just a comma, e.g., s.split(":") will split on a colon and s.split("gat") will split on the string "gat".*	"this,old,man".split(",") == ["this", "old", "man"]
' '.join(lst)	concatenate elements of lst, a list of strings, separated by ' ' or any string	':'.join(['a','b','c']) == "a:b:c"

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

List index, splicing and concatenation (+)		
`lst =[3, 6, 8, 1, 7]`		
		Example
lst[x]	index an element	lst[0] == 3 lst[-1] == 7
lst[x:y]	splice of list, sublist from index x up to but not including index y	lst[1:3] == [6, 8] lst[:4] == [3, 6, 8, 1] lst[3:] == [1,7]
+ operator	concatenate two lists	[3,4] + [1,3,2] == [3,4,1,3,2]

List Functions		
`lst =[3, 6, 8, 1, 7]`		
sum(lst)	returns sum of elements in list `lst`	sum([1,2,4]) == 7
max(lst)	returns maximal element in `lst`	max([5,3,1,7,2]) == 7
lst.append(...)	append an element to lst, changing lst	[1,2,3].append(8) == [1,2,3,8]

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.