CPS305 Lab 1

Topics:

- Recursion

Files:

- Lab1.py
- Incorrectly named files will receive a mark of zero.

Submit Files:

- Submit file through D2L
- Your file should not have any print statements in it.
- You will lose a mark for every print statements.
- There is a getName() function in the Lab1.py file.
- Change the output string to your last and first name.
- Your name should be EXACTLY as written on D2L.
- Incorrectly filed out getName function result in a mark of zero.

Lab Description:

This problem is inspired by a one of the problems in CCP109 recursion problem. If you had no problems with recursion problems in 109, this should be a cakewalk.

Completing Lab 1 will demonstrate an understanding of using recursive algorithms to solve problems that needs branching to find the solution.

Given a number 'n', and a list of prime numbers 'p_list' (in ascending order), find a sum that use some combination of numbers in p_list that add up to 'n.' For example if n = 17 and p_list = [2, 3, 5], one sum is 2+5+5+5 = 17.

You must write two functions sum_exists and find_sum. Both of these functions arguments that includes a number 'n' and list of primes 'p_list.'

```
def sum_exists(n, i = 0, p_list)
```

The prototype for sum_exists is given above. The function should return True if there exists a combination of numbers from p_list that add up to 'n' and False if none exists. This function **MUST** be implemented using **recursion**.

n	P_list	Expected Result
17	[2, 3, 5]	True
7	[3, 5]	False
107	[2, 3, 37]	True
43	[3, 29]	False
146	[17, 29, 37]	True

```
def find_sum(n, i = 0, p_list, sum_list)
```

The prototype for find_sum is given above. The function should return an expression of a sum of numbers from p_list that add up to 'n' in the form of a list. If no sum exists, return an empty list. You do not have to find all of the sums, just the first one that you find. This function **MUST** be implemented using **recursion**.

n	P_list	Expected Result
17	[2, 3, 5]	[2, 2, 2, 2, 2, 2, 2, 3]
7	[3, 5]	[]
107	[[2, 3, 37]	[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3]
43	[3, 29]	[]
146	[17, 29, 37]	[17, 17, 17, 29, 29, 37]

This lab can be solved using brute force. Of course, we do not want that and thus, when you run the tester, it will timeout after a while and if your lab does not finish you would not get the marks.

Testing Your Code:

You can test your code with the tester. The tester is not a be all and end all. You should rely on your own testing before using the tester. Note that the lab should be done on Python 3 and the tester only works on Python 3. The tester is named Lab1Tester.py. Your file must be named correctly and be in the same folder of the tester. Here is an example of executing it on the command line (note that the $ is representative of the beginning of the new line).

```
$ python3 Lab1Teser.py
```