CPS305 Lab2

Topics:

- Queues and Stacks

Files:

- Lab2.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:

In this lab, you are resolving Lab 1. However, you must do so using an iterative method. To solve for it with an iterative method, you will need to use a stack ADT. You must implement a Stack ADT using a linked solution. Detail implementation requirements are as follow

Requirements:

- First, input your name in the getName() method. If this does not work, you will receive a mark of zero.
- A Node class is given. You are to create a Stack ADT using a linked implementation. The implementations are linked Nodes.
- A class called MyStack is given. You are to implement the methods push(), pop(), top() and the over loaded __len__() method. Push should add an item to the stack. Pop should remove an item from the stack. Top should return the data of the top item of the stack but does not remove the item. The function ___len__() should return the number of items in the stack. All methods MUST be O(1).
- You must write a function sum_exists() that behaves exactly like the sum_exists function from Lab 1. The prototype for sum_exists is given below. 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 an **iterative** method that utilize a Stack.

```
def sum_exists(n, p_list)
```

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

Restrictions:

- 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.
- You cannot use Python's list in your implementation. This includes the methods appends, pop, etc. You must use the Node class given.
-

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 Lab2Tester.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 Lab2Teser.py
```