CMPSC-122: Intermediate Programming
Spring 2018

Lab #10

Due Date: 03/30/2018, 11:59PM

Instructions:

- The work in this lab must be completed alone.
- If you need guidance, attend to your recitation class.
- Read the "Submitting assignments to Vocareum" file for instructions on how to submit this lab
- Do not change the function names or given code on your script
- The file name must be LAB10.py (incorrect name files will get a 0 score)
- You are responsible for testing your code. Use `python -i LAB10.py` in your terminal (or command prompt) to provide input to your functions. Test with as many data as you feel comfortable
- Each function must return the output (Do not use print in your final submission)
- Do not include test code outside any function in the upload. Remove all your testing code before uploading your file. If you are using input() to insert values in your functions and print to see the values, remove them.

Exercise 1 [10 pts]

NOTE: There is no partial credit for methods that do not work properly. You must ensure the entire code works after the addition of new methods.

In class, we discussed the abstract data type Stack. A stack is a collection of items where items are added to and removed from the top (LIFO). Implement the stack data structure with the following operations:

- *Stack*() creates a new stack that is empty. It needs no parameters and returns nothing
- push(item) adds a new Node with value=item to the top of the stack. It needs the item and returns nothing. [2 pts]
- pop() removes the top Node from the stack. It needs no parameters and returns the value of the Node removed from the stack. Modifies the stack. [2 pts]
- peek() returns the value of the top Node from the stack but does not remove it. It needs no parameters. The stack is not modified. [2 pts]
- isEmpty() tests to see whether the stack is empty. It needs no parameters and returns a boolean value. [2 pts]
- size() returns the number of items on the stack. It needs no parameters and returns an integer [2 pts]

EXAMPLE

```
>>> stack=Stack()
>>> stack.push(2)
>>> stack.push(4)
>>> stack.push(6)
```

```
>>> stack.printStack()
6
4
2
>>> stack.pop()
6
>>> stack.printStack()
4
2
>>> stack.size()
2
>>> stack.isEmpty()
False
>>> stack.push(15)
>>> stack.printStack()
15
4
2
>>> stack.peek()
15
>>> stack.printStack()
15
4
2
>>> stack.top
15
```

Tips:

- Starter code contains the method printStack()
- Make sure you update the pointers top and next according to the operation performed