Lab 3

Exercise 1

Show how to implement a Queue using two stacks?

```
"""How do you implement a queue using 2 stacks"""
from Labs import Stack


class QueueFromStack:
    def __init__(self):
        self.main_stack = Stack.Stack()
        self.temp_stack = Stack.Stack()

    def enque(self, item):
        # items added to the end will simply be
appended
        self.main_stack.push(item)

    def deque(self):
        # pops items to a temporary stack to reverse
the order (now ordered as a list)
        for i in range(self.main_stack.size()):
            self.temp_stack.push(self.main_stack.pop())

        # saves the first element to be popped
        out = self.temp_stack.pop()

        # returns the elements back to the original
```

```
stack to allow for safe enqueing
        for j in range(self.temp_stack.size()):
            self.main_stack.push(self.temp_stack.pop())

        return out

    def is_empty(self):
        return len(self.main_stack.stack) == 0

    def size(self):
        return self.main_stack.size()

#instantiating
myQueue = QueueFromStack()

#populating the queue
myQueue.enque(1)
myQueue.enque(2)
myQueue.enque(3)
myQueue.enque(4)
myQueue.enque(5)

#popping (FIFO order)
while not myQueue.is_empty():
    print(myQueue.deque())
```

Output

```
1
2
3
```

```
4
5
```

- we enque in the order 1,2,3,4,5 and pop in the same order so we have FIFO functionality

Exercise 2

```
Assume that you are not as careful programmer as you
are and you frequently forget to close your  brackets
while you program – (, {,... You find it difficult to
find where the mistake is. Write a  program that reads
in a sequence of characters, and determines whether its
parentheses, braces,  and curly braces are "balanced."
Hint: for left delimiters, push onto stack; for right
delimiters, pop  from stack and check whether popped
element matches right delimiter.
```

```
"""This Functionn will take in a series of
encapsulating charecters and return
true if they are balanced
"""
from Labs import Stack

def is_balanced(val: str):
    s = Stack.Stack()

    # we will iterate over a list of chars in val
    for v in list(val):
        if v == "(":
```

```
            s.push(")")
        elif v == "[":
            s.push("]")
        elif v == "{":
            s.push("}")

        # check for right delimiters
        else:
            # if we have a right delimiter and the
stack is empty...we have too many right delimiters
            if s.is_empty():
                return False

            elif s.pop() != v:
                return False

    #if we run through the loop and there are items in
the stack...not enough right delimeters
    if not s.is_empty():
        return False

    return True

test = "[(])"
print(is_balanced(test))
```

Output

```
False
```

- as we have a case where a right delimiter doesn't match a left delimeter the function returns false