**Assignment 2**

Python 3.6.6

**Main.py**

Function read\_stack\_input

variables:

mystack – MyStack data structure

push\_count – int

pop\_count – int

max\_count – int

output\_file – file to print results to

input\_file – file to receive instructions from

for line in file

read the line and figure out with method to call

pass the value if push is called

increment the appropriate counter

add all the counts together

write output to the output\_file

close output\_file

function read\_queue\_input

variables:

myqueue – MyQueue data structure

enqueue\_count – int

dequeue\_count – int

output\_file – file to print results to

input\_file – file to receive instructions from

for line in file

read the line and figure out what method to call

pass the value if enqueue is called

increment the appropriate counter

add all the counters together

write output to output\_file

close output\_file

**node.py**

Class Node

function\_\_init\_\_:

being passed in: node\_data – value being passed in; previous\_node – pointer to last

node; next\_node -pointer to next node

Self.node\_data – value

Self.previous\_node – pointer

Self.next\_node - pointer

Function is\_head

Return true if the previous node is empty

False if it isn’t

Function is\_tail

Return true if the next node is empty

False if it isn’t

**First\_queue.py**

Class Queue

Function \_\_init\_\_

Self.first – Node

Self.last – Node

Self.size – Int

Function enqueue

Param: new value for node

Create the node

Check if there is a first node

If not,

assign first and last to the new node

If there is,

Connect the last node and the new node

Make the new node the last node

Increment size

Function dequeue

Assign first node to a temp node

If the node exists

Set the next node to the first node

Decrement size

Return the dequeued node

Function is\_empty

Return true if there isn’t a first node

Function size

Return the size of the queue

**First\_stack.py**

Class Stack

Function \_\_init\_\_

Self.top – node on top of stack

Self.size – int

Function push

Param: new\_node – value to create new node with

If stack is empty

Add node to the top of stack and that’s it

Else:

Connect the top and the new node

Set the top to the new node

Increment the size

Function pop

Set temp node to top

If that node isn’t empty

Set the top to the tops previous node

Disconnect the nodes

Decrement size

Return the temp node

Function is\_empty

Return true if the top node is empty

Function size

Return the size of the stack

**myQueue.py**

Class MyQueue

Function \_\_init\_\_

Self.first – first node

Self.last – last node

Self.size – int

Self.stack\_one – stack data structure

Self.stack\_two – stack data structure

Function enqueue

Param: new\_node – value to add the the queue

If stack\_one is empty

Add the first node to the stack

If not

Pop stack\_one to stack\_two

Add the new node to stack\_one

Pop stack\_two back onto stack\_one

Increment the size

Function dequeue

pop stack\_one into a temp variable

assign a new first node

decrement the size

return the dequeued\_node

**myStack.py**

class MyStack

function \_\_init\_\_

self.top – top node

self.size – int

self.queue\_one – Queue data structure

self.queue\_two - Queue data structure

self.max\_stack – stack\_data structure

function push

param: new\_node – new value to add the stack

check if there is a first node

if not,

add a new node to queue\_one

if there is a first node

dequeue queue\_one into queue\_two

add the new queue into queue\_one

check if the new value is a max value

dequeue queue\_two into queue\_one

set a new top node

increment size

function pop

dequeue queue\_one into a temp variable

assign a new top node

pop the max\_stack

decrement the size

return the dequeued\_node

function is\_empty

return true if the top node is empty

function size

return the size of the stack

function get\_max

return the top of the max\_stack

function discover\_if\_new\_max

param: new node being added to the stack

if the new node is greater than the top of the stack

push the new node into the max\_stack

if the new node is less than the top of the stack

push the value of the top onto the max\_stack