Name:_____

In Class Activity #5 – Stacks and Queues

1. Looking at the ADT description of a Stack data structure write some code to (2pts)
    - create a stack object
    - add the string “hello” to the stack
    - add the string “bye” to the stack
    - remove the top item in the stack and print it out – what do you expect to be printed?
    - print out the size of the stack object – what do you expect to be printed?

2. Draw a picture of the stack from the example on the slides (1 pt)

3. The following is an implementation of a Stack using a list with **top at index = 0**. Update the code to implement use **top at index = len-1**. Hint: only 3 lines need to be updated. (2 pt)

```
class Stack:
 def __init__(self):
    self._items = []

   def is_empty(self):
      return len(self._items) == 0

   def push(self, item):
     self.items.insert(0,item)

   def pop(self):
    return self.items.pop(0)

   def peek(self):
    return self.items[0]

   def size(self):
     return len(self._items)
```

4. State the Big-O time complexity of each Stack operation when top is at index=len-1 in terms of the size of the list n. (2 pts)

```
push(item)
pop()
peek()
is_empty()
size()
```

5. Looking at the ADT description of a Queue data structure write some code to (2pts)
    - create a Queue object
    - add the string “hello” to the Queue
    - add the string “bye” to the Queue
    - remove the top item in the Queue and print it out
    - print out the size of the Queue object

6. Draw a picture to show the Queue from the example on the slides and what is printed (1 pt)

7. The code below shows a Queue using lists where the head is at len-1 and tail is at 0. Update the code so that head is at index 0 and tail is at len-1. Hint: Only 2 lines need update. (2 pts)

```
class Queue:
    def __init__(self):
       self._items = []

    def is_empty(self):
        return len(self._items) == 0

    def enqueue(self, item):#add to tail
       self._items.insert(0,item)

    def dequeue(self):#remove from head
       return self._items.pop()

    def size(self):
      return len(self._items)
```

8. State the Big-O time complexity of each Queue operation when head is at index=0, and tail at index len -1 in terms of the size of the list n. (2 pts)

```
enqueue(item)
dequeue()
is_empty()
is_empty()
size()
```