In the following, we investigate two data structures—a queue and a stack—constructed using an underlying data structure like a Python-style list. Hint: Test your code in python before submitting this homework.

1. A queue is a container (like a list) that holds values. There are two important operations: enqueue and dequeue. enqueue is a procedure that adds a value to the queue. dequeue removes and returns that element of the queue that was added first. It is a first-in, first-out or FIFO ("fife-oh") structure. If we wait in line, that’s a queue: the person waiting the longest is the next to get attention.

   a. Inside the queue class, we’ll use a Python list to hold the queue values:

   ```python
   class queue(object):
     __slots__ = ['_lst']
     def __init__(self):
       """Initialize the queue."""
       self._lst = []
   
   Write the enqueue method for this class:
   ```
   ```python
   def enqueue(self,v):
     """Add an element to a queue; it will be 'last in line'.""
   ```

   b. Write the dequeue method for this class. Remember: it removes and returns the longest-waiting value from the queue.

   ```python
   def dequeue(self):
     """Remove and return longest-waiting value in the non-empty queue.""
   ```

   c. Write the __str__ method for this class. __str__ is called when you print(...) a queue.

   ```python
   def __str__(self):
     """Returns a string representation of this class.""
   ```
2. A stack is a container that also offers two operations: push and pop. push is a procedure that adds a value to the stack. pop is a function that removes and returns that element of the stack that was added last. It is a last-in, first-out or LIFO ("life-oh") structure. Dining hall trays are stored in a stack: the tray on the top is used frequently while the tray at the bottom is rarely used.

a. Inside the stack class, we’ll use a tuple object to hold the stack values:

```python
class stack(object):
    __slots__ = ['_tup']
    def __init__(self):
        """Initialize the stack."""
        self._tup = ()

Write the push method for this class:

    def push(self,v):
        """Add v to the stack."""
```

b. Write a method, isEmpty, for this class. It is a property of the stack that is True if the stack is empty, and False otherwise:

```python
@property
    def isEmpty(self):
        """Return True iff the stack is empty."""
```

c. Write the pop method for this class, returning the popped value:

```python
    def pop(self):
        """Remove and return value most recently added to the non-empty stack."""
```