Faculty of Mathematical Economics

Data Structures and Algorithms

Instructor: **Nguyen Thanh Tuan**

DSEB Class of 2021 - 2024

Homework Assignment Week 6

Topic: Queue

Date Created: February 23, 2023

Problem 1: Queue Implementation With Limited Capacity

a. Implement the `Queue` class that you learned in the class again with below structure:

```python
class Queue:
    """FIFO Queue implementation using a Python list as underlying storage."""

    def __init__(self):
        """Create an empty queue"""
        pass

    def __len__(self):
        """Return the number of elements in the queue"""
        pass

    def is_empty(self):
        """Return True if the queue is empty"""
        pass

    def __repr__(self):
        """Return string representation of the queue"""
        pass

    def first(self):
        """Return (but do not remove) the element at the front of the queue.

        Raise Empty exception if the queue is empty.
        """
        pass

    def dequeue(self):
        """Remove and return the first element of the queue.

        Raise Empty exception if the queue is empty.
        """
        pass

    def enqueue(self, e):
        """Add an element to the back of queue"""
        pass
```

b. Modify the Queue implementation above so that the queue's capacity is limited to maxlen elements, where maxlen is an optional parameter to the constructor (that defaults to None). If enqueue is called when the queue is at full capacity, throw a Full exception (defined similarly to Empty).

```python
class QueueMaxLen:
    """FIFO Queue implementation with limited capacity using a Python
   list as underlying storage."""

    # your code here
```

Problem 2: QueueByStack and StackByQueue

a. Using `Stack` from previous class (or using class Stack below) to implement `Queue`:

```python
class Stack:
 """LIFO Stack implementation using a Python list as underlying storage
   """

    def __init__(self):
    """Create an empty stack"""
        self._data = [] # nonpublic list instance

    def __len__(self):
    """Return the number of elements in the stack."""
        return len(self._data)

    def is_empty(self):
    """Return True if the stack is empty."""
        return len(self._data) == 0

    def push(self, e):
    """Add element e to the top of the stack."""
        self._data.append(e) # new item stored at end of list

    def top(self):
    """Return (but do not remove) the element at the top of the stack.

    Raise Empty exception if the stack is empty.
    """
        if self.is_empty():
            raise Empty("Stack is empty")
        return self._data[-1] # the last item in the list

    def pop(self):
    """Remove and return the element from the top of the stack (i.e.,
   LIFO).
```

```

    Raise Empty exception if the stack is empty.
    """
        if self.is_empty():
            raise Empty("Stack is empty")
        return self._data.pop() # remove last item from list
```

b. Using `Queue` from Problem 1.a. to implement `Stack`.

Problem 3: Buy Tickets for Blackpink Concert

There are n people in a line queuing to buy tickets for Blackpink concert, where the 0^{th} person is at the front of the line and the $(n-1)^{th}$ person is at the back of the line.

You are given a 0-indexed integer array tickets of length n where the number of tickets that the i^{th} person would like to buy is tickets[i].

Each person takes exactly 1 second to buy a ticket. A person can only buy 1 ticket at a time and has to go back to the end of the line (which happens instantaneously) in order to buy more tickets. If a person does not have any tickets left to buy, the person will leave the line.

Implement a function using `Queue` to return the time taken for the person at position k (0-indexed) to finish buying tickets.

Note: To get full mark at this problem, you had better use `tickets` as `Queue`

See the examples below:

```
tickets = [2,3,2]
k = 2
get_time(tickets, k)
>>> 6
 """Explanation:
- In the first pass, everyone in the line buys a ticket and the line
   becomes [1, 2, 1].
- In the second pass, everyone in the line buys a ticket and the line
   becomes [0, 1, 0].
The person at position 2 has successfully bought 2 tickets and it took
   3 + 3 = 6 seconds.
 """

tickets = [5,1,1,1]
k = 0
get_time(tickets, k)
>>> 8
 """Explanation:
 - In the first pass, everyone in the line buys a ticket and the line
   becomes [4, 0, 0, 0].
```

```
- In the next 4 passes, only the person in position 0 is buying tickets
    .
The person at position 0 has successfully bought 5 tickets and it took
    4 + 1 + 1 + 1 + 1 = 8 seconds."""
```

Guidelines for submission

- Your submission must be under the `.ipynb` format.
- Your submission will be graded and it is likely that homework grade will contribute as a component in your GPA.
- If your submission is later than the due date without special consideration approval, you will receive a penalty on your mark.