****

**College of Engineering, Software Engineering Department**

**SE 2231/36 Algorithms**

**Laboratory 2**

**Deques and Randomized Queues**

**Objectives:**

* Implement a deque and a randomized queue.

Write a generic data type for a deque and a randomized queue. The goal of this assignment is to implement elementary data structures using arrays and linked lists, and to introduce you to generics and iterators. Make sure you’re using Python version 3.12+.

**Dequeue.** A double-ended queue or deque (pronounced “deck”) is a generalization of a stack and a queue that supports adding and removing items from either the front or the back of the data structure. Create a generic data type Deque that implements the following API using a **doubly-linked list**:

# remember that Item is a generic variable, you declare it

# when making an object

#

# For example:

# d = Deque[int]()

#

***class*** Deque[Item]:

    # construct an empty deque

***def*** **\_\_init\_\_**(**self**):

**pass**

    # is the deque empty?

***def*** **is\_empty**(**self**) -> *bool*:

**pass**

    # return the number of items on the deque

***def*** **size**(**self**) -> *int*:

**pass**

    # add the item to the front

***def*** **add\_first**(**self**, **item**: Item) -> None:

**pass**

    # add the item to the back

***def*** **add\_last**(**self**, **item**: Item) -> None:

**pass**

    # remove and return the item from the front

***def*** **remove\_first**(**self**) -> Item:

**pass**

    # remove and return the item from the back

***def*** **remove\_last**(**self**) -> Item:

**pass**

***def*** **\_\_iter\_\_**(**self**):

**pass**

    # return the current item and tick the current item to the next

    # otherwise, raise StopIteration

***def*** **\_\_next\_\_**(**self**):

**pass**

    # unit testing (required)

**@***staticmethod*

***def*** **main**():

**pass**

*Corner cases.* Throw the specified exception for the following corner cases:

* Raise a ValueError if the client calls either add\_first() or add\_last() with a None argument.
* Raise an IndexError if the client calls either remove\_first() or remove\_last() when the deque is empty.

*Unit testing.* Your main() method must call directly every method to help verify that they work as prescribed (e.g., by printing results to standard output).

**Randomized queue.** A *randomized queue* is similar to a stack or queue, except that the item removed is chosen uniformly at random among items in the data structure. Create a generic data type RandomizedQueue that implements the following API using a list:

***class*** RandomizedQueue[Item]:

    # construct an empty randomized queue

***def*** **\_\_init\_\_**(**self**):

**pass**

    # is the randomized queue empty?

***def*** **is\_empty**(**self**) -> *bool*:

**pass**

    # return the number of items on the randomized queue

***def*** **size**(**self**) -> *int*:

**pass**

    # add the item

***def*** **enqueue**(**self**, **item**: Item) -> None:

**pass**

    # remove and return a random item

***def*** **dequeue**(**self**) -> Item:

**pass**

    # return a random item (but do not remove it)

***def*** **sample**(**self**) -> Item:

**pass**

    # for looping this object will loop over the items in a random order

***def*** **\_\_iter\_\_**(**self**):

**pass**

    # return the current item and tick the current item to the next

    # otherwise, raise StopIteration

***def*** **\_\_next\_\_**(**self**) -> Item:

**pass**

    # unit testing (required)

**@***staticmethod*

***def*** **main**():

**pass**

*Corner cases.* Throw the specified exception for the following corner cases:

* Raise a ValueError if the client calls enqueue() with a None argument.
* Raise an IndexError if the client calls either sample() or dequeue() when the randomized queue is empty.

*Unit testing.* Your main() method must call directly every method to verify that they work as prescribed (e.g., by printing results to standard output).

**Client.** Write a client program permutation.py that takes an integer k as a command-line argument; reads a sequence of strings from a .txt file and prints exactly k of them, uniformly at random. Print each item from the sequence at most once.

**Web submission.** Submit a .zip file containing only deque.py, randomized\_queue.py and permutation.py . Deadline: February 24, 2025, 11:59pm

Submit to <https://www.dropbox.com/request/57rCIOiGOf1YrpSvPFBv>.