Project 2: Deques

Due: Friday Feb 09, 11:59 pm

1 Assignment Overview

For this project you will be implementing a double-ended queue (or deque). Your deque will be used to store and manipulate date from a given text file. In addition to the standard deque methods, you will be implementing three bulk methods for modifying the whole deque.

2 Assignment Deliverables

You must turn in completed versions of the following files:

- Deque.py

Be sure to use the specified file name and to submit your files for grading via **D2L Dropbox** before the project deadline.

3 Assignment Specifications

Your task will be to complete the methods listed below:

- `__len__`
- `peek_front`
- `peek_back`
- `push_front`
- `push_back`
- `pop_front`
- `pop_back`
- `clear`
- `retain_if`
- `__iter__`

Your implementation of the first seven methods should run in $O(1)$ amortized time. `clear` and `retain_if` may run in $O(n)$ time. `iter` can take up to $O(n)$ time total to yield all of the items (each individual item should be produced in amortized constant time). Your deque must use $O(n)$ space.

The `peek` and `pop` methods should raise a `IndexError` if the deque is empty. No other exceptions should be thrown.

You should include comments where appropriate. In particular, you should describe the overall method used to implement your deque.

4 Assignment Notes

- Points will be deducted if your solution has warnings of any type.
- You have been supplied with stubs for the required methods. You must complete these methods to complete the project. You may add more functions than what was provided, but **you may not modify the signatures of the provided methods**.
- You do not have to worry about error checking for valid input. You may assume that the supplied reader function correctly reads the input file.
- You **do** have to worry about accessing elements from an empty deque.
- Implementations for `is_empty` and `repr` have been provided. Note that these rely on the `len` and `iter` functions, respectively, so you may want to complete these functions early.
- You have been provided with some unit tests that can be used to assess your solution. There are additional test cases that will be kept hidden.
- It is your responsibility to check for potential edge cases. The supplied unit tests do not account for all possible valid inputs
- The `condition` parameter in the `retain_if` method is a function pointer. You can invoke it with `condition(e)` just like you would for a regular function.
- For the `iter` method, you may want to use the `yield` keyword.
- You may not use any of the classes in the `collections` module.