UNIVERSITY OF THE PHILIPPINES VISAYAS
COLLEGE OF ARTS AND SCIENCES
DIVISION OF PHYSICAL SCIENCES AND MATHEMATICS

CMSC 123
Data Structures
A.Y. 2022 – 2023

Assignment Guide

Prepared by:

Jayvee B. Castañeda
Instructor

ACADEMIC INTEGRITY

As a student of the University of the Philippines, I pledge to act ethically and uphold the value of honor and excellence. I understand that suspected misconduct on given assignments/examinations will be reported to the appropriate office and if established, will result in disciplinary action in accordance with University rules, policies and procedures. I may work with others only to the extent allowed by the Instructor.

Laboratory Exercise #4: Advanced Queues

For this laboratory exercise, you are tasked to implement the Queue ADT in an object-oriented programming approach using Python.

Using the Array and DLL implementation from your Laboratory Exercise #1 and #2, your task is to implement the Circular Queue ADT and the Deque ADT using Array and DLL approaches respectively. Each one has its own set of attributes and methods only applicable to themselves. You are required to use ONLY the attributes and/or methods given as much as possible. ***You are NOT allowed to add new attributes and methods outside of the ones given. You are also not allowed to use Python's built-in functions.***

CIRCULAR QUEUE

Attributes	**Function**
size	stores the number of items in the queue
contents []	stores the elements in the queue
frontIndex	stores the index of the first element in the queue
rearIndex	stores the index of the last element in the queue
Methods	**Function**
front()	returns a reference value to the front element of the queue, but doesn’t remove it
*enqueue(***value***)*	inserts an element at the end of the queue
dequeue()	removes the item at the front of the queue, and returns the element that was removed

DLL DEQUE

Attributes	**Function**
size	stores the number of nodes in the deque
headNode	stores the value of the front node of the deque
tailNode	stores the value of the rear node of the deque
Methods	**Function**
first()	returns a reference value to the front node of the deque , but doesn’t remove it
last()	returns a reference value to the rear node of the deque , but doesn’t remove it
*insertFirst(***value***)*	inserts a node at the front of the deque

*insertLast(***value***)*	inserts a node at the end of the deque
removeFirst()	removes the node at the front of the deque, and returns the node that was removed
removeLast()	removes the node at the end of the deque, and returns the node that was removed

BREAKDOWN OF POINTS

	Points
Circular Queue	50
DLL Deque	50
TOTAL	**100**

If you wish to learn more about object-oriented programming in Python, you may check out the links below:

- https://www.w3schools.com/python/python_classes.asp
- https://www.geeksforgeeks.org/python-oops-concepts/
- https://realpython.com/python3-object-oriented-programming/

Submission Instructions

- This activity should **ONLY** be done using the computers in the Laboratory Rooms during our class schedule.
- A good programming practice is to write comments on important line of codes for readability and documentation. ***Documentation is required for this laboratory exercise.***
- You are required to **finish and defend your code personally as soon as possible before the deadline specified by the instructor** in our classroom.
- You are only allowed to defend during class hours. ***Late submissions will have deductions*.** All laboratory exercises should be submitted on or before the last day of classes.

Good luck! 😊

- *Sir Jayvee*

Rubrics for Programming Exercises					
		Excellent	**Good**	**Fair**	**Poor**
Program Code (50%)	**Correctness** (10%)	**9-10** Program displays correct output with no errors	**6-8** Program displays output with minor errors	**3-5** Program displays output with multiple errors	**0-2** Program displays incorrect output
	Logical Structure (15%)	**11-15** Program is logically well-designed	**7-10** Program has slight logic errors that do not significantly affect the results	**4-6** Program has significant logic errors	**0-3** Program has incorrect logic
	Elegance/ Standardization (10%)	**9-10** Program is stylistically well designed	**6-8** Program has few inappropriate design choices (i.e., poor variable names, improper indentation)	**3-5** Several inappropriate design choices (i.e., poor variable names, improper indentation)	**0-2** Program is poorly written
	Readability (10%)	**9-10** Program is easily readable and understandable	**6-8** Program is mostly readable with few confusing parts	**3-5** Program is only slightly readable with most parts confusing	**0-2** Program is barely or not readable nor understandable

	Documentation **(5%)**	**5** **Program is well documented**	**4** **Program is missing one required comment**	**2-3** **Program is missing two or more required comments**	**0-1** **Program has little or no documentation**
Code Defense **(50%)**	**Mastery** **(25%)**	**20-25** **Student has comprehensive knowledge of his/her code**	**14-19** **Student has some knowledge of his/her code with minor inconsistencies**	**7-13** **Student has few knowledge of his/her code with major inconsistencies**	**0-6** **Student has little or no knowledge of his/her code**
	Understanding **(25%)**	**20-25** **Student can answer all questions given by the instructor with full understanding**	**14-19** **Student can answer some questions given by the instructor with minor inconsistencies**	**7-13** **Student can answer few questions given by the instructor with major inconsistencies**	**0-6** **Student can only answer little or no questions given by the instructor**