ASHESI UNIVERSITY

CS222: Data Structures and Algorithms
Fall Semester (Aug - Dec 2023)
Lab Exercise 4 (Stack, Queue, Deque)

STATEMENT ABOUT ACADEMIC HONESTY AND INTEGRITY

Academic honesty and integrity are very important at Ashesi and central to the achievement of our mission: To train a new generation of ethical and entrepreneurial leaders in Africa to cultivate within our students the critical thinking skills, concern for others, and the courage it will take to transform a continent. As this mission is our moral campus, we recommend you take it seriously in this course without any exceptions at all.

Ashesi therefore does not condone any form of academic dishonesty, including plagiarism and cheating on tests and assessments, amongst other such practices. Ashesi requires students to always do their own assignments and to produce their own academic work unless given a group assignment.

As stated in Ashesi's student handbook, Section 7.4:

"Academic dishonesty includes plagiarism, unauthorized exchange of information or use of material during an examination, unauthorized transfer of information or completed work among students, use of the same paper in more than one course, unauthorized collaboration on assignments, and other unethical behaviour. Disciplinary action will be taken against perpetrators of academic dishonesty."

All forms of academic dishonesty are viewed as misconduct under Ashesi Student Rules and Regulations. Students who make themselves guilty of academic dishonesty will be brought before the Ashesi Judicial Committee and such lack of academic integrity will have serious consequences for your academic records.

INSTRUCTIONS:

1. This is an individual assignment. So, every student must independently work and submit.
2. This lab assignment contains *two* questions. You are required to solve both of them.
3. Question1 carries 80 Marks and Question2 carries 20 Marks.
4. You are required to show the execution of your program as a screen-recorded video (max 5 min) and have to answer the reflective/oral question to prove your knowledge and understanding of the concepts.
5. Your solution program should contain comments explaining statements that involve important computations.
6. You need to upload your original '.java' file, screen-recorded video, and a document (pdf/Word) containing an answer to the reflective/oral question given.

7. The deadline for submission: 30 October 2023, 12 PM.

PROBLEM 1: Task Management System.

The objective of this lab exercise is to implement a basic task management system using stacks **only**. Therefore you need to thoroughly understand the workings of stack and queue data structures.

Task Description: Imagine you are building a task management system for a small team. In this system, tasks can be added, removed, and processed. The tasks are added to the system when they are created, and they are processed in the order they are created, normally. However, the system should also consider the priority of tasks and the high-priority tasks are to be processed first always. Once a task is processed, it can be removed from the system alerting the user. If the user still wants to keep the processed tasks, a separate list of processed tasks user wish to maintain should be available. Write a Java program that simulates the task management system as described.

You are expected to work on this task as follows.

1) *Create a Task Class*: Define a class called Task that has the following attributes:

- ✓ task_id (a unique identifier for each task).
- ✓ description (a description of the task).
- ✓ status (can be "pending" or "completed").
- ✓ priority (an integer indicating the task priority, 0-5 indicates **normal** priority, 5-10 indicates **high** priority)

2) *Implement a Stack*: Create a stack data structure to store tasks. The stack is used to maintain and process the high priority tasks. Remember that the high priority task should be always processed first (e.g., task with priority 9 should be processed earlier than the task with priority 6 though the task with priority 6 is created first). The stack class may contain the following methods.

- ✓ push(task): Add a task to the stack.
- ✓ pop(): Remove and return the top task from the stack.
- ✓ top(): Return the top task without removing it.

3) *Implement a Queue*: Create a queue data structure to process the normal tasks. Implement the following queue operations:

- ✓ enqueue(task): Add a task to the end of the queue.
- ✓ dequeue(): Remove and return the task from the front of the queue.

Additional instructions: Your system should be a menu-driven interactive application that allows the user to add tasks to the stack or queue as required, process the tasks as described, and view the current stack and queue. After processing tasks, if user wishes to keep, move them to a separate queue changing their status to "completed." Implement a simple text-based user interface that displays the menu options and allows the user to interact with the task management system. The menu should include options like "Add Task," "Process Task," "View Stack," "View Queue," and "Quit." (For interactive menu idea, see the programs on Stack and Queue implementation demonstrated in class and shared on WhatsApp and CANVAS).

Test your task management system by adding tasks, processing them, and checking if the stack and queue behave as expected.

Summary: Ensure that normal tasks are processed in the order they were added, priority tasks are processed in the order of their priority, and list of processed tasks should be maintained as per the user wish.

Marking:

Submission/Attempt: 20 Marks, Execution as per the task: 40 Marks, Comments within the code explaining the program statements: 20 Marks.

PROBLEM 2: Reflection and Alternative solution

Write a reflection on how using stacks and queues in this task management system can be helpful in real-world scenarios. Particularly, your reflection should address the following questions.

Are there any issues concerning the fair treatment of the tasks?

Do we need stack and queue both to solve the given problem?

How are such systems built usually in practice? What data structures they use?

Is there any possible best solution still using stacks and queues and for fair treatment on task processing?

Can the application be built using an alternative data structure such as Deque alone instead of stack and queue? If so, describe the changes you would make referring to the line numbers in your code of the problem number 1 above.

Marking:

Issues on fair process treatment and reflection on the whole application building experience considering the real-world solutions: 10 Marks

Description of alternative solution using Deque: 10 Marks.