Mid-term Exam of DSA-Lab
Total Marks:40
V2

Work with templates

Task 1:

Create an abstract class Queue with the following attributes and functions:

Attributes:

- int front
- int rear
- int maxSize
- Type* arr (An array to store queue elements)

Pure Virtual Functions:

- Enqueue: Add an element to the queue.
- Dequeue: Remove an element from the queue.
- Peek: View the front element of the queue without removing it.
- Display: Display all elements in the queue.
- Size: Return the current size of the queue.
- Empty: Check if the queue is empty.
- Full: Check if the queue is full.

Constructor and Destructor:

- Implement a constructor and destructor for the abstract class.

Now, create a function called AvgReplacement(myQueue&, K) which takes the queue object myQueue and an integer K as input. The function replaces every K-th element of the queue with the average of the K preceding elements (if there are at least K elements). Perform this operation inside the main function.

Input 1:

{1, 2, 3, 4, 5, 6}, K = 3

Output:

1 2 3 4 4 6

Input 2:

{2, 4, 5, 2, 3, 6}, K = 3

Output:

2 4 5 4 3 6

Input 3:

{5, 3, 2, 6}, K = 2

Output:

5 3 4 6

The AvgReplacement function replaces every K-th element with the average of the previous K elements in the queue.

Task 2:

You are designing a **Library Management System** to handle two key operations efficiently:

Book Stack (LIFO):

1. Books returned by readers are added to a **stack**.
2. The librarian processes the most recently returned book first (LIFO).
2.

Reader Queue (FIFO):

1. Readers requesting assistance form a **queue**.
2. Readers are served in the order they arrive (FIFO).

As the developer, your task is to implement these functionalities using **singly linked lists** while adhering to the following constraints:

Constraints

1. Use a singly linked list to implement both the stack and the queue.
2. Only a **head pointer** is allowed to manage the linked list.
3. No additional pointers (like tail) or built-in libraries are permitted.

The library's return counter uses a stack to manage books. You need to implement the following operations for the **Stack**:

void push(int value):

1. Add a book with its ID (integer) to the stack.

int pop():

1. Remove and return the ID of the most recently returned book.
2. If the stack is empty, display an appropriate message.

int peek():

1. View the ID of the most recently returned book without removing it.

2. If the stack is empty, display an appropriate message.

bool isEmpty():

1. Check if the stack is empty.

The library's assistance counter uses a queue to manage readers. You need to implement the following operations for the **Queue**:

void enqueue(int value):

1. Add a reader with their ID (integer) to the queue.

int dequeue():

1. Remove and return the ID of the next reader to be served.
2. If the queue is empty, display an appropriate message.

int front():

1. View the ID of the next reader without removing it.
2. If the queue is empty, display an appropriate message.

bool isEmpty():

1. Check if the queue is empty.

---

1. A librarian manages the return counter with the following operations:
    1. Add books with IDs **101**, **102**, and **103** to the stack (push).
    2. Display the ID of the most recently returned book (peek).
    3. Process two returned books and display their IDs (pop).
    4. Check if the return stack is empty (isEmpty).

1. A librarian manages the assistance counter with the following operations:
    1. Add readers with IDs **201**, **202**, and **203** to the queue (enqueue).

2. Display the ID of the next reader to be served (front).
3. Serve two readers and display their IDs (dequeue).
4. Check if the reader queue is empty (isEmpty).