National University of Computer and Emerging Sciences

Laboratory Manual #6

for

Data Structures Lab

(CL 2001)

Department of Computer Science
FAST-NU, Lahore, Pakistan

Introduction

Objectives

After performing this lab, students shall be able to:

1. Queues
2. Stacks

Problem 1: Implementing a Queue using linked list

Implement a template-based queue using **Linked List**. The required member methods are:

a) **void enqueue()**: Adds an element to queue
b) **void dequeue()**: Removes an element from queue
c) **bool isFull():** return true if queue is full else false.
d) **int size()**: returns the count of total element stored in the stack.
e) **bool isEmpty()**: returns true if the stack is empty else false.
f) **int front()**: returns the element on Front of queue
g) **int rear()**: return the element on Rear of queue

Problem 2: Implementing Queue with stack

You are tasked with implementing a queue data structure using two stacks. You need to create a C++ class StackQueue that simulates a queue using two stacks: s1 for enqueue operations and s2 for dequeue operations. Your task is to complete the StackQueue class and implement the following operations:

- **enqueue(int item):** Add an item to the queue.
- **dequeue():** Remove and return an item from the queue. If the queue is empty, return -1.
- **isEmpty():** Check if the queue is empty. Return true if it's empty, false otherwise.

Problem 3: Implementing a Min Queue

Design a queue data structure that supports the following operations:

a) **enqueue(item):** Add an item to the back of the queue.
b) **dequeue():** Remove and return the item from the front of the queue.
c) **getMin():** Retrieve the minimum element from the queue in constant time.

Ensure that the **getMin** operation provides the minimum element efficiently without iterating through the entire queue every time.

Problem 4: Bank Transaction Simulation

You are tasked with simulating a simple bank transaction system using stacks and queues. The bank has several teller windows, and customers can perform various transactions, including withdrawals, deposits, and checking balances. Your task is to implement this system.

Create a C++ program to simulate the bank transactions. Here are the details:

Requirements:

1. Use a queue to represent the line of customers waiting to be served at the bank. Each customer is represented by a dictionary with the following properties:
    - customer_id: A unique identifier for each customer.
    - transaction_type: One of the following: "withdrawal," "deposit," or "check_balance."
    - amount: The amount involved in the transaction (for withdrawals and deposits).
    - balance: The current account balance for checking balance transactions.
2. Use a stack to represent each teller's transaction history. Each teller can handle one customer at a time, and the stack should keep track of the customer's transaction history.

Operations:

- A customer arrives at the bank and joins the queue.
- When a teller is available, they pop the next customer from the queue and serve them.
- The teller performs the requested transaction, updates the customer's balance, and pushes the customer's data onto the teller's stack.
- If a customer requests a "check_balance" transaction, the teller provides the balance without modifying it.
- After the transaction is complete, the customer leaves the bank.

Your Program:

Implement the following functions:

1. **enqueue_customer(queue, customer_data)**: Add a customer to the queue.
2. **teller_serve_customer(queue, teller_stack):** Serve the next customer in the queue. Perform the transaction and update the customer's data.
3. **check_balance(queue, teller_stack, customer_data):** Check the customer's account balance without modifying it.
4. **get_transaction_history(teller_stack):** Retrieve the transaction history for a teller's current customer.