INHA UNIVERSITY IN TASHKENT
School of Computer and Information Engineering
FALL SEMESTER 2025

Project- (Students Learning)

DATA STRUCTURES

M.M: 20 Marks Submission Date: **3 DECEMBER 2025**

Banking System with Transaction History (Linked List + Queue + Tree)

Problem Description

This project simulates a simple banking system where multiple customers hold accounts.
The system allows:

- Creating a new bank account
- Depositing or withdrawing money
- Maintaining a daily queue of transactions
- Keeping a transaction history
- Searching accounts using efficient data structures

It demonstrates how **Data Structures** can be used to manage financial operations in a real system.

Data Structures

A. Binary Search Tree (BST) → Store Account Information

Each bank account is stored as a node in a BST. The account number is used as the **key**.

BST allows:

- **Fast search**
- **Fast insertion**
- **Fast sorting by account number**

Structure of each BST node:

```
Account Number
Account Holder Name
Balance
Pointer to transaction history (Linked List)
Left Child
Right Child
```

B. Linked List → Transaction History

Every account maintains a **linked list of transactions**.

Each node represents one transaction:

```
Type: Deposit / Withdrawal
Amount
Date & Time
Next pointer
```

Benefits of using Linked List

- Transactions grow dynamically.
- Easy to insert at end.
- No need for fixed-size storage.

C. Queue → Pending Transactions

Every bank has many transactions pending (ex: during peak hours).

Use a **queue** to store them:

- Transactions are processed **first come, first served (FIFO)**.
- Each queue node contains:
- `Account Number`
- `Type (Deposit/Withdraw)`
  `Amount`

Features

1. Create Account

- User enters account number, name, and initial balance.
- A new BST node is created.
- Inserted at the correct position according to the BST rules.

2. Deposit Money

Two ways:

1. **Direct deposit** → update account balance + add to transaction history.
2. **Queued deposit** → added to daily queue and processed later.

History entry example:

```
Deposited 500 on 14-11-2025 10:24 AM
```

3. Withdrawal

Similar to deposit:

- Check sufficient balance.
- If insufficient → show error.
- Else deduct and record in the linked list history.

History entry example:

```
Withdrawn 200 on 14-11-2025 11:05 AM
```

4. Daily Transaction Queue

All scheduled or automatically generated transactions go into a queue.

Processing the queue:

- DEQUEUE one transaction
- Find the correct account using BST search
- Perform deposit/withdraw
- Insert the action into the transaction linked list

This simulates a real bank server processing tasks in the background.

5. Display Transaction History

Given an account number:

- Search BST
- Access its linked list
- Print all transactions in order

6. Search Account

Search using BST:

- Very efficient for large number of accounts.
- Displays:
  - Account number
  - Name
  - Balance
  - Total number of transactions

Extensions (Optional But Recommended)

Add Interest Calculator

Add a function to:

- Take annual interest rate
- Apply monthly or yearly interest to all accounts
- Add a transaction record like:

B. Use Hashing for Faster Lookup

Instead of BST, create a **hash table**:

- Account number → index in an array
- Much faster than BST for huge data sets
- Good for demonstrating collision handling (chaining / open addressing)