# CS 338 Fun Assignment #5

## Question 1.

Based on the given database scenario and concurrent transactions, determine the appropriate isolation levels. Consider an online banking system with the following database schema:

- Account(account_id, customer_id, balance, account_type)
- Transaction_Log(log_id, account_id, transaction_type, amount, timestamp)

For each of the following transaction scenarios, determine the **lowest appropriate isolation level** and explain your reasoning:

**(a)** Transaction T1 performs a single account balance update:

```
-- T1:
UPDATE Account
SET balance = balance + 500
WHERE account_id = 12345;
COMMIT;
```

**(b)** Transaction T2 generates a monthly account statement:

```
-- T2:
SELECT customer_id, balance, account_type
FROM Account
WHERE customer_id = 67890;

SELECT log_id, transaction_type, amount, timestamp
FROM Transaction_Log
WHERE account_id IN (
    SELECT account_id FROM Account WHERE customer_id = 67890
);
COMMIT;
```

**(c)** Transaction T3 calculates and updates interest for all savings accounts:

```
-- T3:
SELECT account_id, balance
FROM Account
WHERE account_type = 'SAVINGS';

-- For each account found, calculate 2% interest
UPDATE Account
SET balance = balance * 1.02
WHERE account_type = 'SAVINGS';

-- Log the interest transactions
INSERT INTO Transaction_Log
```

```
SELECT NEXTVAL('log_seq'), account_id, 'INTEREST', balance * 0.02, NOW()
FROM Account
WHERE account_type = 'SAVINGS';
COMMIT;
```

## Question 2.

Conceptual questions on concurrency anomalies and storage search method.

**(a)** Explain the difference between an **unrepeatable read** and a **phantom read**.

**(b)** Suggest 1 **benefits** and 1 **potential problems** of maintaining secondary indexes in a high-update OnLine Transaction Processing workload (systems that handle a large number of short, fast transactions in real time, onine banking for example).

## Question 3.

Design and analyze a database system that handles both transaction isolation and indexing requirements. You are designing a library management system with the following tables:

- Book(isbn, title, author, category, copies_available)
- Member(member_id, name, email, membership_type)
- Checkout(checkout_id, member_id, isbn, checkout_date, due_date, returned_date)

**(a)** The system needs to handle the following concurrent transaction. Determine the minimum isolation level required.

A member checks out a book

```
-- T_CHECKOUT:
SELECT copies_available FROM Book WHERE isbn = '978-0123456789';
-- If copies_available > 0:
UPDATE Book SET copies_available = copies_available - 1
WHERE isbn = '978-0123456789';
INSERT INTO Checkout
VALUES (NEXTVAL('checkout_seq'), 12345, '978-0123456789',
        CURRENT_DATE, CURRENT_DATE + 14, NULL);
COMMIT;
```

Consider that multiple members might try to check out the last copy of the same book simultaneously.

**(b)** Consider the following two queries that are frequently executed on the library management system:

T1:

```
SELECT * FROM Book WHERE isbn = '978-0123456789';
```

T2:

```
SELECT b.title, b.author, c.checkout_date, c.due_date
FROM Book b, Checkout c
WHERE b.isbn = c.isbn
  AND c.returned_date IS NULL
  AND c.member_id = 12345;
```

For each query, design an optimal indexing strategy.