	<b>School of Engineering &amp; Technology</b>	
	<b>Department: CSE</b>	<b>Session: Odd</b>
	<b>Programme:- B.Tech CSE</b>	<b>Semester: 1</b>
	<b>Course Code: ETCCCP105</b>	<b>Number of students:</b>
	<b>Course Name: Computer Science Fundamentals &amp; Career Pathways</b>	<b>Faculty: Dr. Ravinder Beniwal</b>

## Assignment Number 01

Foundations of Computer Science & Computational Thinking

### Assignment Title

*Design and Simulate a Real-World Process Using Flowcharts and Pseudocode*

Name – Aditya Bauthiyal

Roll no. – 2501010303

Section – E

## **Introduction**

### Problem Statement - Library Book Borrowing System

A library borrowing system allows users to borrow and return books efficiently.

The system checks member credentials, verifies book availability, and updates the library database accordingly. It helps automate the book lending process, maintain accurate records, and ensure that borrowing limits and due dates are properly managed.

This program will help to understand the algorithms and logical thinking by using concepts like decomposition, abstraction, pattern recognition to also apply in real world to solve problems, flowchart and pseudocode to plan systems effectively.

## **PROBLEM ANALYSIS**

Problem Statement :- Library Book Borrowing System

a) **Abstraction** – focusing only on essential elements while ignoring the non-essential ones.

- System verifies member status
- User searches for a book
- System checks if the book is available
- If available → issue book and update records
- Log in with member ID
- If not available → display “Book not available”

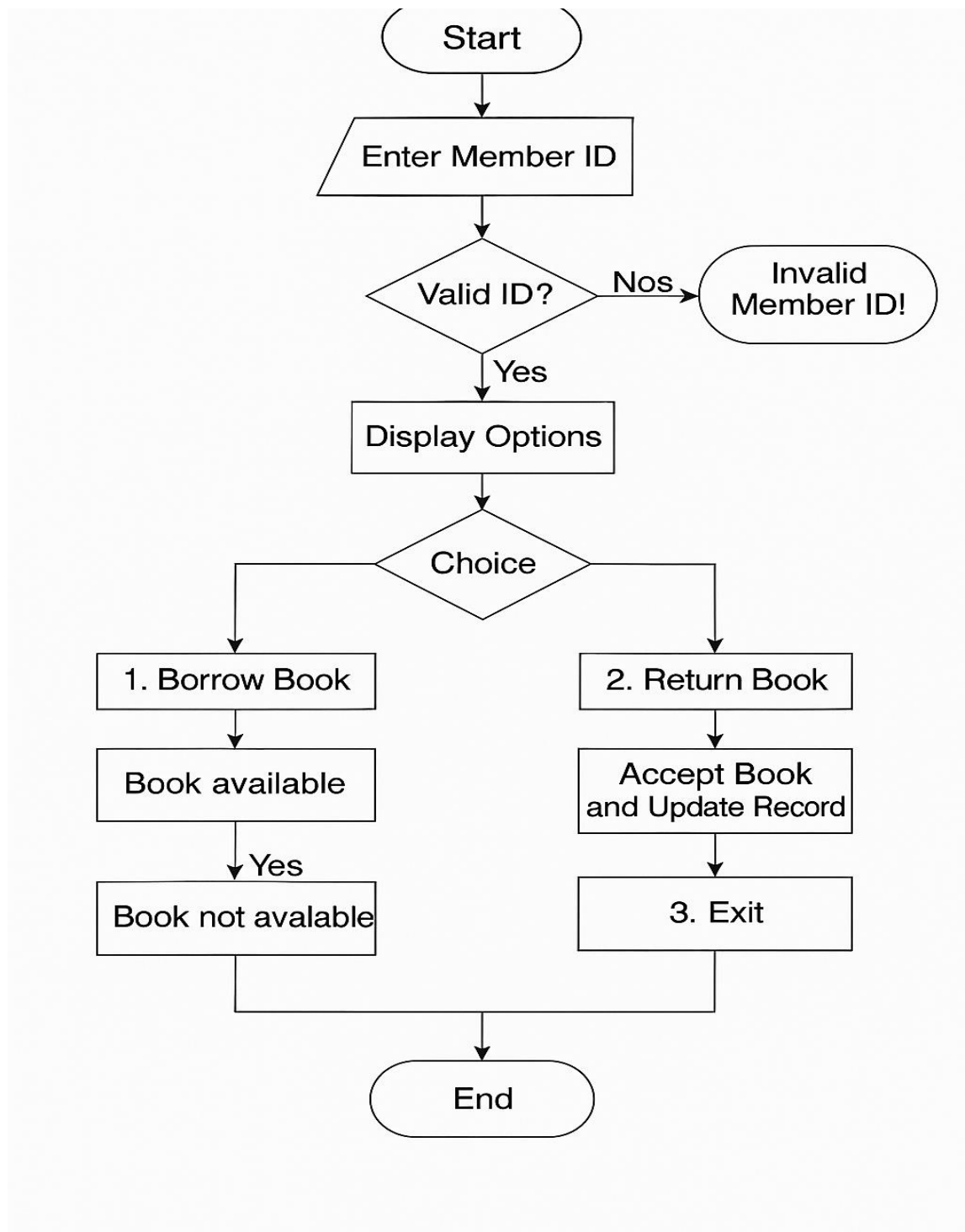
b) **Decomposition** – breaking the problem into smaller components.

- Input: Member ID, book title/ID.
- Verification: Validate member and check borrowing limit.
- Search: Locate book in catalog.
- Borrowing: Issue book and update records.
- Return: Accept returned book and update status.
- Error Handling: Display messages for invalid input, unavailable books, or exceeded limits.

c) **Pattern recognition** – looking for similarities and repeated patterns to predict the solution to the problem.

- Repeated user verification for each transaction.
- Similar checks for borrowing and returning actions.
- Updating database consistently for every change.

## Design – Flowchart and pseudocode



## Pseudocode

BEGIN LIBRARY\_SYSTEM

SET max\_books  $\leftarrow$  3

SET borrowed\_books  $\leftarrow$  0

SET books\_available  $\leftarrow$  ["Book1", "Book2", "Book3"]

DISPLAY "Enter Member ID"

INPUT member\_id

IF is\_valid\_member(member\_id) = TRUE THEN

    DISPLAY "1. Borrow Book"

    DISPLAY "2. Return Book"

    DISPLAY "3. Exit"

    INPUT choice

    IF choice = 1 THEN

        IF borrowed\_books < max\_books THEN

            DISPLAY "Enter Book Name to Borrow: "

            INPUT book\_name

            IF book\_name IN books\_available THEN

                REMOVE book\_name FROM books\_available

                borrowed\_books  $\leftarrow$  borrowed\_books + 1

                DISPLAY "Book issued successfully!"

            ELSE

                DISPLAY "Book not available."

            ENDIF

        ELSE

```

        DISPLAY "Borrowing limit reached!"

    ENDIF

ELSE IF choice = 2 THEN

    DISPLAY "Enter Book Name to Return: "

    INPUT book_name

    ADD book_name TO books_available

    borrowed_books ← borrowed_books - 1

    DISPLAY "Book returned successfully!"

ELSE IF choice = 3 THEN

    DISPLAY "Thank you! Visit again."

ELSE

    DISPLAY "Invalid choice."

ENDIF

ELSE

    DISPLAY "Invalid Member ID!"

ENDIF

END

-----

FUNCTION is_valid_member(member_id)

    SET valid_member_ids ← ["KRMU1", "KRMU2", "KRMU3"]

    IF member_id IN valid_member_ids THEN

        RETURN TRUE

    ELSE

        RETURN FALSE

    ENDIF

END FUNCTION

```

## **Implementation of code with outputs**

```
def library_system():
    max_books = 3
    borrowed_books = 0
    books_available = ["Book1", "Book2", "Book3"]

    member_id = input("Enter Member ID: ")

    if is_valid_member(member_id):
        print("1. Borrow Book")
        print("2. Return Book")
        print("3. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            if borrowed_books < max_books:
                book_name = input("Enter Book Name to Borrow: ")

                if book_name in books_available:
                    books_available.remove(book_name)
                    borrowed_books += 1
                    print("Book issued successfully!")
                else:
                    print("Book not available.")
            else:
                print("Borrowing limit reached!")

        elif choice == 2:
            book_name = input("Enter Book Name to Return: ")
            books_available.append(book_name)
            borrowed_books -= 1
            print("Book returned successfully!")

        elif choice == 3:
            print("Thank you! Visit again.")

        else:
            print("Invalid choice.")
```

```
    else:
        print("Invalid Member ID!")

def is_valid_member(member_id):
    valid_member_ids = ["KRMU1", "KRMU2", "KRMU3"]
    return member_id in valid_member_ids

library_system()
```

PROBLEMS 135

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS C:\Users\Dell\OneDrive\Desktop\python> & C:/Users/Dell/AppData/
y
Enter Member ID: M001
1. Borrow Book
2. Return Book
3. Exit
Enter your choice: 1
Enter Book Name to Borrow: Book2
Book issued successfully!
PS C:\Users\Dell\OneDrive\Desktop\python>
PS C:\Users\Dell\OneDrive\Desktop\python> █
```

PROBLEMS 135

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS C:\Users\Dell\OneDrive\Desktop\python> & C:/Users/Dell/A
y
Enter Member ID: M12
Invalid Member ID!
PS C:\Users\Dell\OneDrive\Desktop\python> █
```

PROBLEMS 135

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS C:\Users\Dell\OneDrive\Desktop\python> & C:/Users/Dell/AppD
y
Enter Member ID: M003
1. Borrow Book
2. Return Book
3. Exit
Enter your choice: 2
Enter Book Name to Return: Book2
Book returned successfully!
PS C:\Users\Dell\OneDrive\Desktop\python> █
```