

A. Objective: Implement matrix multiplication using multithreading.

Requirements:

1. Write a program to multiply two matrices A and B.
2. Use multithreading to calculate each element of the resulting matrix C concurrently, where
$$C[i][j] = \sum (A[i][k] \times B[k][j])$$
3. Each thread should compute a single element of matrix C.
4. Use input validation to ensure that the matrices can be multiplied (i.e., the number of columns in A matches the number of rows in B).
5. Display the resulting matrix after computation.

Marking Scheme:

- Multithreading implementation (10 marks).
- Correct matrix multiplication logic (10 marks).
- Input validation and output formatting (5 marks).

B. Library Management System with OOP

Objective: Implement an object-oriented solution for a library system.

Requirements:

1. Create a Book class:

- **Attributes:** bookID (String), title (String), author (String), availableCopies (int).
- **Methods:**
 - borrowBook(): Reduces the availableCopies by 1 if copies are available, otherwise throws a custom exception BookNotAvailableException.
 - returnBook(): Increases the availableCopies by 1.

2. Create a Library class:

- **Attributes:** A list of Book objects.
- **Methods:**
 - addBook(Book book): Adds a book to the library.
 - searchBookByTitle(String title): Returns the book object if found.
 - displayAvailableBooks(): Prints details of books with at least one available copy.

3. Custom Exception:

- Implement BookNotAvailableException that triggers when a book is not available to borrow.

4. Main Class:

- Add some books to the library.
- Perform borrowing, returning, and searching for books.
- Handle exceptions gracefully.

Marking Scheme:

- OOP Design (5 marks).
- Implementation of Book and Library classes (10 marks).
- Custom Exception handling (5 marks).
- Correct functionality and testing (5 marks).

C. Objective: Manage student records using file handling and OOP principles.

Requirements:

1. Create a Student class:

- **Attributes:** rollNumber (int), name (String), marks (double).
- **Methods:**
 - toString(): Returns a string representation of the student data.

2. File Operations:

- Write student data to a file named students.txt.
- Read student data from the file and display it on the console.

3. Main Class:

- Take input for multiple students (at least 5) and write their data to the file.
- Read the file content and display all student details.
- Add functionality to search for a student by roll number and display their details.

Marking Scheme:

- OOP Design (5 marks).
- File writing and reading functionality (10 marks).
- Student search implementation and input handling (5 marks).
- Correct execution and output formatting (5 marks).