INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Smart Hostel Management System

Design and implement a **Smart Hostel Management System** in C++ using `if`, `for`, `switch`, arrays, and pointers (without using recursion, structures, or classes or functions). The program should provide the following functionalities:

1. **Room Allocation:** Maintain an array of hostel rooms (e.g., 50 rooms). Each element of the array indicates whether the room is vacant (0) or occupied (1). Students should be allocated rooms on a first-available basis.
2. **Search Student by Room Number:** Implement a search feature (linear or binary search) to check if a given room number is allocated and display the student ID if available.
3. **Check Availability:** Count the number of vacant rooms using a `for` loop and display them.
4. **Fee Payment Status:** Maintain an additional array to store fee status (0 = Not Paid, 1 = Paid). Allow updating fee payment when a student pays.
5. **Exit Management:** On student exit, deallocate the room (set room status back to 0).

Input:

- Student ID (integer)
- Room number (integer)
- Fee status (0/1)

Output:

- Room allocation details
- Availability of rooms
- Fee payment status
- Search result for student room

Use a `switch` menu to provide user choices for the above functionalities. The system should run in a loop until the user chooses to exit. Save the output to a file.

```
========== Smart Hostel Management System ==========
1. Allocate a Room to Student
2. Vacate a Room
3. Check Room Availability
4. View Student Details
5. Display All Occupied Rooms
6. Exit
Enter your choice:
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Smart Hospital Management System

Design and implement a **Smart Hospital Management System** in C++ using `if`, `for`, `switch`, arrays, and pointers (without using recursion, structures, or classes). The program should provide the following functionalities:

1. **Patient Registration:** Maintain an array of hospital beds (e.g., 50 beds). Each element of the array indicates whether the bed is vacant (0) or occupied (1). Patients should be admitted to the first available bed.
2. **Search Patient by Bed Number:** Implement a search feature (linear search) to check whether a given the bed number is allocated and displays the patient ID (e.g. MA25M0xx) if available.
3. **Check Bed Availability:** Count the number of vacant beds using a `for` loop and display them.
4. **Treatment Status:** Maintain an additional array to store treatment status for patients (0 = Ongoing, 1 = Completed). Allow updating treatment status when a patient is discharged.
5. **Discharge Patient:** On discharge, free the bed (set bed status back to 0) and update treatment completion.

   Input:

   - Patient ID (integer)
   - Bed number (integer)
   - Treatment status (0/1)

   Output:

   - Bed allocation details
   - Availability of beds
   - Treatment status of a patient
   - Search result for patient by bed number

   Use a `switch`-case menu to provide user choices for the above functionalities. The system should run in a loop until the user chooses to exit. Save the output to a file.

```
========== Smart Hospital Management System ==========
1. Admit a Patient
2. Discharge a Patient
3. Check Bed Availability
4. View Patient Details
5. Display All Occupied Beds
6. Exit
Enter your choice:
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Smart Library Management System

Objective: Design and implement a C++ program to manage the operations of a library using arrays and pointers. The system should allow users to perform various tasks such as issuing books, returning books, searching for a book, and checking availability. The solution should be implemented using only basic constructs such as `if`, `for`, `switch`, arrays, and pointers (no functions, structures, or recursion).

Description: The library contains a fixed collection of 200 books. Each book is uniquely identified by a **Book ID** ranging from 1 to 200. An array of size 200 will be used where each element represents the availability of a book:

- Value `0` $\rightarrow$ Book is available.
- Value `1` $\rightarrow$ Book is issued.

Tasks to be performed:

1. **Issue a Book:** The user provides a Book ID. If the book is available (`0`), mark it as issued (`1`) and confirm the action. If already issued, display an appropriate message.
2. **Return a Book:** The user provides a Book ID. If the book is currently issued (`1`), mark it as available (`0`). Otherwise, display that the book was not issued.
3. **Search for a Book:** Given a Book ID, check its status and display whether it is available or issued.
4. **Display Availability:** Count and display the total number of available books and issued books in the library.
5. **Exit:** Terminate the program.

Implementation Requirements:

- Use a one-dimensional integer array of size 200 to represent the library book collection.
- Use pointers to traverse and manipulate the array elements.
- Implement the menu using a `switch` statement to allow the user to select different operations.
- Use `if` and `for` loops wherever applicable.
- Do not use structures, classes, recursion, or library functions other than `iostream`.

Save the output to a file.

Sample Menu:

```
1. Issue a Book
2. Return a Book
3. Search for a Book
4. Display Availability
5. Exit
Enter your choice:
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Smart Cinema Ticket Booking System

Objective: Design and implement a C++ program to manage ticket reservations for a cinema hall using arrays and pointers. The system should allow users to book a ticket, cancel a ticket, check the availability of a seat, and display the current seating arrangement. The implementation should strictly use only basic constructs such as `if`, `for`, `switch`, arrays, and pointers (no functions, structures, or recursion).

Description: A cinema hall has 50 seats, numbered 1 to 50. An integer array of size 50 will be used to represent the seat status:

- Value `0` → Seat is available.
- Value `1` → Seat is booked.

Tasks to be performed:

1. **Book a Ticket:** The user provides a seat number (1–50). If the seat is available (`0`), mark it as booked (`1`) and display a confirmation. If the seat is already booked, display an appropriate message.
2. **Cancel a Ticket:** The user provides a seat number. If the seat is currently booked (`1`), mark it as available (`0`). Otherwise, display that the seat was not booked.
3. **Check Seat Availability:** Given a seat number, display whether the seat is available or booked, using linear search.
4. **Display Seating Arrangement:** Print the status of all 50 seats in a formatted way (e.g., 5 rows of 10 seats), showing available and booked seats.
5. **Exit:** Terminate the program.

Implementation Requirements:

- Use a one-dimensional integer array of size 50 to represent the cinema seats.
- Access and update array elements using pointers.
- Implement the menu-driven system using a `switch` statement.
- Use `if` and `for` loops wherever necessary.
- Do not use structures, classes, recursion, or library functions other than `iostream`.

Save the output to a file.

Sample Menu:

```
1. Book a Ticket
2. Cancel a Ticket
3. Check Seat Availability
4. Display Seating Arrangement
5. Exit
Enter your choice:
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Smart Bus Reservation System

Objective: Design and implement a C++ program to manage the seat reservation of a bus using arrays and pointers. The system should allow users to reserve a seat, cancel a reservation, check seat availability, and view the current seating chart. The implementation should strictly use only basic constructs such as `if`, `for`, `switch`, arrays, and pointers (no functions, structures, or recursion).

Description: A bus has 40 seats, each uniquely numbered from 1 to 40. An array of size 40 will be used where each element represents the status of a seat:

- Value `0` → Seat is available.
- Value `1` → Seat is reserved.

Tasks to be performed:

1. **Reserve a Seat:** The user provides a seat number (1–40). If the seat is available (`0`), mark it as reserved (`1`) and display a confirmation. If the seat is already reserved, display an appropriate message.
2. **Cancel a Reservation:** The user provides a seat number. If the seat is currently reserved (`1`), mark it as available (`0`). Otherwise, display that the seat was not reserved.
3. **Check Seat Status:** Given a seat number, display whether it is available or reserved.
4. **Display Seating Chart:** Print the status of all 40 seats in a formatted manner (e.g., 4 rows of 10 seats each), showing which are available and which are reserved.
5. **Exit:** Terminate the program.

Implementation Requirements:

- Use a one-dimensional integer array of size 40 to represent the bus seats.
- Use pointers to access and modify the array elements.
- Implement the menu system using a `switch` statement for user interaction.
- Use `if` and `for` loops wherever necessary.
- Do not use structures, classes, recursion, or library functions other than `iostream`.

Save the output to a file.

Sample Menu:

```
1. Reserve a Seat
2. Cancel a Reservation
3. Check Seat Status
4. Display Seating Chart
5. Exit
Enter your choice:
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Tic-Tac-Toe Game System

Objective: Design and implement a C++ program to simulate a Tic-Tac-Toe game between two players using arrays and pointers. The system should allow players to place their marks (`X` and `O`), check for a winner, and display the current game board. The implementation should strictly use only basic constructs such as `if`, `for`, `switch`, arrays, and pointers (no functions, structures, or recursion).

Description: The game is played on a 3×3 board. A one-dimensional array of size 9 will be used where each element represents a cell on the board:

- Value `0` $\rightarrow$ Empty cell.
- Value `1` $\rightarrow$ Player 1's move (`X`).
- Value `2` $\rightarrow$ Player 2's move (`O`).

Tasks to be performed:

1. **Make a Move:** The current player selects a cell number (1–9). If the cell is empty (`0`), mark it with the player's symbol (`1` for Player 1, `2` for Player 2). If the cell is already occupied, display an appropriate message and let the player try again.
2. **Check Winner:** After each move, check all rows, columns, and diagonals. If three identical marks are aligned, declare the corresponding player as the winner. If all cells are filled without a winner, declare a draw.
3. **Display Game Board:** Print the current status of the board in a 3×3 grid, showing `X` for Player 1, `O` for Player 2, and `.` for empty cells.
4. **Restart Game:** Reset the array so all cells become empty (`0`), and start a new game.
5. **Exit:** Terminate the program.

Implementation Requirements:

- Use a one-dimensional integer array of size 9 to represent the board.
- Use pointers to access and modify the array elements.
- Implement the menu system using a `switch` statement for user interaction.
- Use `if` and `for` loops wherever necessary.
- Do not use structures, classes, recursion, or library functions other than `iostream`.

Save the output to a file.

Sample Menu:

```
Make a Move

Check Winner

Display Game Board

Restart Game

Exit
Enter your choice:
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 4 **MA517M-Basic Programming Laboratory** 8 September 2025
Name **Roll No.: MA25M**

Problem Statement: Smart 2048 Game with WASD Controls

Step 1: Play the 2048 game from this website https://2048game.com/ for 10 minutes. After playing, read the following statements and recreate this game using C++.

Objective: Design and implement a C++ program to simulate the classic 2048 game on a 4×4 grid using arrays and pointers. The system should allow the player to slide tiles using WASD keys, merge equal numbers, and generate new tiles until the player either wins by reaching 2048 or loses when no moves are possible. The implementation should strictly use only basic constructs such as `if`, `for`, `switch`, arrays, and pointers (no functions, structures, or recursion).

Description: The game board is a 4×4 grid (16 cells). A one-dimensional array of size 16 will be used to represent the cells:

- Value `0` $\rightarrow$ Empty cell.
- Any positive integer $(2, 4, 8, 16, \dots)$ $\rightarrow$ Tile value.

At the start of the game, two random tiles (either 2 or 4) are placed on the board. The player can move tiles in four directions using WASD keys:

- `W` $\rightarrow$ Move Up
- `A` $\rightarrow$ Move Left
- `S` $\rightarrow$ Move Down
- `D` $\rightarrow$ Move Right

When two tiles with the same value collide, they merge into a single tile with double the value. After every valid move, a new tile (2 or 4) spawns at an empty position.

Tasks to be performed:

1. **Move Tiles:** The player enters `W`, `A`, `S`, or `D`. All tiles slide in that direction, merging equal tiles where applicable.
2. **Spawn New Tile:** After each valid move, generate a new tile (2 or 4) in a random empty cell.
3. **Display Grid:** Print the 4×4 grid, showing the values of the tiles. Empty cells should be displayed as a dot (`.`).
4. **Check Game Status:**
   - If any cell contains 2048, display a winning message.
   - If the grid is full and no moves are possible, display a "Game Over" message.

5. **Exit:** Allow the player to quit the game by pressing `Q`.

Implementation Requirements:

- Use a one-dimensional integer array of size 16 to represent the 4×4 grid.
- Use pointer arithmetic to access and modify grid elements.
- Use `if`, `for`, and `switch` statements wherever necessary.
- **Only `iostream` is allowed — no external libraries, classes, or recursion.**

Save the output to a file.

Sample Menu:

```
Enter your move (W=Up, A=Left, S=Down, D=Right, Q=Quit):
```