CSC107 2.0 Computer Programming - Laboratory I
Lab sheet 13

Note: Use **multi-dimensional arrays** appropriately.

1. Complete the following C program for a 2D array of size 3x3, to print the matrix.

```
int main() {
    int matrix[3][3] = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };
    printf("Matrix:\n");
    for ……. {
        for …….. {
            printf("%d ", ……);
        }
        printf(".....");
    }
    return 0;
}
```

2. Write a program in C for a 2D array of size 4x4 to input and print the matrix values.

3. Write a C program to perform the subtraction of two matrices (m x n) and store the result in a third matrix. The user should input the elements of both matrices.

4. Write a C program to compare two matrices (m x n). The program should check if the matrices are equal and print an appropriate message.

5. Write a C program to input and print a 3D array with dimensions 4x4x4. The program should

    a. Prompt the user to input values for each element in the 3D array.
    b. Print the values in a formatted manner, displaying each 2D slice of the 3D array separately.

    Sample output:

```
Enter values for the 3D array (4x4x4):
Enter value for array[0][0][0]: 1
Enter value for array[0][0][1]: 2
Enter value for array[0][0][2]: 3
Enter value for array[0][0][3]: 4
...
Enter value for array[3][3][3]: 64
```

```
The 3D array is:
Slice 0:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16

Slice 1:
17 18 19 20
21 22 23 24
25 26 27 28
29 30 31 32

Slice 2:
33 34 35 36
37 38 39 40
41 42 43 44
45 46 47 48

Slice 3:
49 50 51 52
53 54 55 56
57 58 59 60
61 62 63 64
```

6. Write a C program to create a seating arrangement for a hall with 10 rows and 15 seats per row. The program should ensure that participants are seated with one empty seat between each occupied seat. Display the final seating arrangement. Hint available seats in O and blocked seats in X.

7. Write a C program to create a 5x5 matrix representing a simple grid. Populate the grid with sequential numbers starting from 1 in a zigzag pattern, where each row alternates between left-to-right and right-to-left. Display the matrix.

8. Write a C program that initializes a 10x10 game board to simulate a simple game where a player ('P') must navigate to find an enemy ('E'). The player starts at the top-left corner of the board, while the enemy is located at the bottom-right corner. The player can move up, down, left, or right using the keys 'U', 'D', 'L', and 'R', respectively. The game continues until the player reaches the enemy. Display the game board after each move and show a message when the player finds the enemy.
   Detailed implementation
   a. Game Board Initialization
      Create a 10x10 grid representing the game board.

Initialize all cells in the grid with the character '.' to denote empty spaces.
Place the player ('P') at the top-left corner of the grid (position [0][0]).
Place the enemy ('E') at the bottom-right corner of the grid (position [9][9]).

b. Game Loop
Continuously display the current state of the game board.
Prompt the user to enter a move using 'U' (up), 'D' (down), 'L' (left), or 'R' (right).
Update the player's position on the board based on the move input:
'U' moves the player up by one cell.
'D' moves the player down by one cell.
'L' moves the player left by one cell.
'R' moves the player right by one cell.
Ensure the player does not move outside the bounds of the board.
Update the board to reflect the player's new position.

c. Game End Condition:
The game ends when the player's position coincides with the enemy's position.
Display a message indicating that the player has found the enemy and the game is over.

Sample output

```
. P . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . E

Enter your move (U: up, D: down, L: left, R: right): D
```