2D Array Practice Questions

📘 Theoretical Questions (Moderate to Tough)

1. In a row-major ordered 2D array of size 3x4, what is the memory address formula to access the element at row `i` and column `j`, assuming base address `B` and element size `s`?
   a) B + s * (j * rows + i)
   b) B + s * (i * columns + j)
   c) B + s * (i + j)
   d) B + s * (columns * i - j)

2. Which of the following is **true** about column-major implementation of a 2D array?
   a) Elements are stored column by column.
   b) Elements are stored row by row.
   c) Columns must be square.
   d) It does not support dynamic memory allocation.

3. For a 2D array of size `m x n`, what is the total memory occupied (in bytes), if each element is of size 4 bytes?
   a) m + n
   b) m × n × 4
   c) m × n
   d) 4m + 4n

4. If a 2D array is declared as `int arr[5][10]`, which access pattern results in **better cache performance** in C/C++?
   a) Column-wise access
   b) Random access
   c) Row-wise access
   d) Diagonal access

5. What is the primary difference between compile-time and run-time initialization of 2D arrays?
   a) Compile-time uses dynamic memory, run-time uses static
   b) Compile-time uses nested loops
   c) Compile-time values are hard-coded; run-time values are input by user
   d) No difference

6. Which of the following is **false** about zero-based indexing in 2D arrays?

   a) It simplifies address calculations

   b) It's used by most programming languages

   c) Index (0,0) refers to the last element

   d) First element is at index (0,0)

7. Given a 2D array stored in row-major order, which access pattern is **least efficient**?

   a) arr[i][j] in nested row→column loop

   b) arr[j][i] in column→row loop

   c) arr[i][j] in row→column loop

   d) arr[i][j] using flattening

8. When modifying a memory address formula for **1-based indexing**, how is the general formula adjusted in row-major?

   a) Subtract 1 from both row and column indexes

   b) Add 1 to base address

   c) Divide total address by 2

   d) Use modulus with base

9. Which of the following is the **correct formula** for calculating the address in column-major order?

   a) B + s * (j * m + i)

   b) B + s * (i * n + j)

   c) B + s * (i + j)

   d) B + s * (j + i)

10. Why is it important to understand the memory layout of 2D arrays?

    a) To make arrays readable

    b) To determine array size

    c) For accurate memory address calculation and performance optimization

    d) To access elements alphabetically

💻 Coding Questions (Moderate to Tough)

11. Write a program in C++ to input a 3x3 matrix and print the sum of all elements in the principal diagonal.

```cpp
```

```
#include<iostream>
using namespace std;
int main() {
    int arr[3][3], sum = 0;
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            cin >> arr[i][j];
    for(int i=0;i<3;i++)
        sum += arr[i][i];
    cout << "Sum: " << sum;
    return 0;
}
```

12. Implement a Python function that checks if a given 2D matrix is symmetric.

python

```
def is_symmetric(matrix):
    n = len(matrix)
    for i in range(n):
        for j in range(n):
            if matrix[i][j] != matrix[j][i]:
                return False
    return True
```

13. Write a program in Java to perform transpose of a 2D array using a temporary array.

java

```
public class TransposeMatrix {
    public static void main(String[] args) {
        int[][] arr = {{1, 2}, {3, 4}};
        int[][] trans = new int[2][2];
        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 2; j++)
                trans[j][i] = arr[i][j];
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++)
                System.out.print(trans[i][j] + " ");
            System.out.println();
        }
    }
}
```

14. In C, write a function to search for an element in a 2D array using linear search and return its position.

```
c

#include<stdio.h>
void search(int arr[3][3], int key) {
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            if(arr[i][j] == key) {
                printf("Found at (%d, %d)\n", i, j);
                return;
            }
    printf("Not found\n");
}
```

15. Implement a program in Python to rotate a 2D square matrix 90 degrees clockwise without using extra space.

```
python

def rotate(matrix):
    n = len(matrix)
    for i in range(n):
        for j in range(i+1, n):
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]
    for row in matrix:
        row.reverse()
```

16. Write a C++ program to dynamically allocate a 2D array of size `m x n`, take user input, and display the matrix.

```
cpp

#include<iostream>
using namespace std;
int main() {
    int m, n;
    cin >> m >> n;
    int** arr = new int*[m];
    for(int i=0;i<m;i++)
        arr[i] = new int[n];
    for(int i=0;i<m;i++)
        for(int j=0;j<n;j++)
            cin >> arr[i][j];
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++)
            cout << arr[i][j] << " ";
        cout << endl;
    }
    for(int i=0;i<m;i++)
        delete[] arr[i];
    delete[] arr;
    return 0;
}
```

17. Given a 2D array, write a Java method to print only the boundary elements in clockwise order.

```java
public static void printBoundary(int[][] arr) {
    int m = arr.length, n = arr[0].length;
    for (int i = 0; i < n; i++) System.out.print(arr[0][i] + " ");
    for (int i = 1; i < m; i++) System.out.print(arr[i][n-1] + " ");
    for (int i = n-2; i >= 0; i--) System.out.print(arr[m-1][i] + " ");
    for (int i = m-2; i > 0; i--) System.out.print(arr[i][0] + " ");
}
```

18. Write a Python function that flattens a 2D array into a 1D list (row-wise).

```python
def flatten(matrix):
    return [elem for row in matrix for elem in row]
```

19. Implement a C program to calculate the sum of each row and each column of a 2D array and display the results.

```c
c


#include<stdio.h>
int main() {
    int arr[3][3], rowSum, colSum;
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            scanf("%d", &arr[i][j]);

    for(int i=0;i<3;i++) {
        rowSum = 0;
        for(int j=0;j<3;j++) rowSum += arr[i][j];
        printf("Row %d sum: %d\n", i, rowSum);
    }

    for(int j=0;j<3;j++) {
        colSum = 0;
        for(int i=0;i<3;i++) colSum += arr[i][j];
        printf("Col %d sum: %d\n", j, colSum);
    }
    return 0;
}
```

20. Write a Python function that takes a 2D array and returns True if all rows and columns are sorted in increasing order.

```python
python


def is_sorted(matrix):
    for row in matrix:
        if row != sorted(row): return False
    for col in zip(*matrix):
        if list(col) != sorted(col): return False
    return True
```

✅ Answer Key (Theoretical Questions)

Q#	Answer
1	b
2	a
3	b
4	c
5	c
6	c
7	b
8	a
9	a
10	c