

Online on Pointer

Section: B1+B2

Time: 50 minutes

Problem 1

Write a C program that multiplies two matrices **A** ($m \times n$) and **B** ($n \times q$) using **only pointers**.

- You **must not use any array indexing syntax** ([]).
- Use malloc or calloc to **dynamically allocate** memory for all matrices (A, B, and C)
- You **must free** all allocated memory before the program exits
- All access, traversal, and assignment must be done through **pointer arithmetic only**.

Input:

- Three integers: m, n, and q — dimensions of matrices:
 - Matrix **A** is $m \times n$
 - Matrix **B** is $n \times q$
- Then input the **elements of matrix A** in a $m \times n$ 2D format:
 - m lines, each with n space-separated integers
- Then input the **elements of matrix B** in a $n \times q$ 2D format:
 - n lines, each with q space-separated integers

Output :

1. Print the size of the resulting matrix **C**:
2. Then print matrix **C = A × B**

Sample 1

Input:

```
2 3 2
1 2 3
4 5 6
7 8
9 10
```

11 12

Output:

2 2
58 64
139 154

Sample 2

Input:

2 2 2
1 0
0 1
5 6
7 8

Output:

2 2
5 6
7 8

Sample 3

Input:

2 3 2
0 0 0
0 0 0
1 2
3 4
5 6

Output:

2 2
0 0
0 0

Problem 2

Write C function

```
int *merge_unique(const int *a, size_t n, const int *b, size_t m, size_t
*out_len);
```

This function takes two sorted arrays of integers (a and b) with no duplicates inside each array. It must return a new array containing the **symmetric difference** — elements that appear in **either a or b, but not in both**.

Function Parameters :

- a: pointer to first sorted array of size n
- b: pointer to second sorted array of size m
- out_len: pointer to size_t, to be set to length of resulting array

Return Value:

- A dynamically allocated array of integers that appear in either a or b but not both.
- If the result is empty, return NULL and set *out_len = 0.

Constraints:

- No use of global variables.
- Do not copy arrays into other containers.
- Must access arrays through pointer arithmetic only (*, +), no [].
- Only use <stdio.h>, <stdlib.h>, and <stddef.h>. size_t is a type defined in <stddef.h>

Sample 1

Input:

```
4          // size of the first array
1 2 5 9    // first sorted array
4          // size of the second array
2 6 9 10   // second sorted array
```

Output:

```
4          // size of the resultant array  
1 5 6 10 // sorted resultant array containing unique elements
```

Sample 2**Input:**

```
3  
1 3 5  
3  
2 4 6
```

Output:

```
6  
1 2 3 4 5 6
```

Sample 3**Input:**

```
3  
1 2 3  
3  
1 2 3
```

Output:

```
0
```

Sample 4**Input:**

```
0  
3  
10 20 30
```

Output:

```
3  
10 20 30
```