

## Online on Pointers

**Section: C1+C2**  
**minutes**

**Time: 50**

### **Problem 1: Reflection of a Square Matrix along the Secondary Diagonal**

You are given a 2D square matrix of size **NxN** containing characters. Write a function to reflect the matrix along the **secondary diagonal** using only pointer arithmetic. You must not use array indexing (e.g., `matrix[i][j]` or `matrix[i*N + j]`). All memory accesses must be performed through pointers.

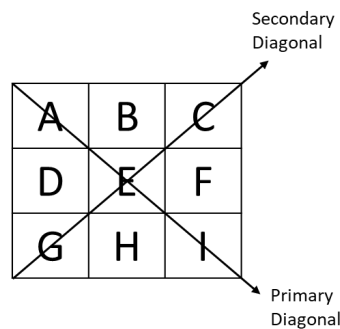
#### **Constraints:**

- $2 \leq N \leq 10$
- Use dynamic memory allocation to initialize the matrix. You **must free** all allocated memory before the program exits.
- You can only use **pointer dereferencing** and **pointer arithmetic**.

#### **Example:**

Input:	Output:
3 a b c d e f g h i	i f c h e b g d a
4 @ @ @ @ @ @ * * @ * @ * @ * * @	@ * * @ * @ * @ * * @ @ @ @ @ @

#### **Note:**



1.

2. Use a **space** before '`%c`' in `scanf()` to handle any newline characters left in the buffer:  
`scanf(" %c", input);`

## Problem 2: Rotate Odd-Positioned Elements in a Sequence

You are given a sequence of integers. Write a function to rotate the elements at **odd positions** (1st, 3rd, 5th, etc.) by a specified number of positions to the **right**. You must use only pointer arithmetic for accessing and modifying the array elements. Do not use array indexing (e.g., `arr[i]`). All memory accesses must be performed through pointers.

### Constraints:

- $1 \leq n, r \leq 10^6$
- Use dynamic memory allocation. You **must free** all allocated memory before the program exits.
- You can only use **pointer dereferencing** and **pointer arithmetic**.

### Input Format:

- The first line contains two integers, **n** and **r**: the number of elements in the sequence and the number of positions to rotate.
- The second line contains **n** integers, representing the elements of the sequence.

### Example:

Input:	Output:
9 1 1 2 3 4 5 6 7 8 9	1 8 3 2 5 4 7 6 9
12 8 -1 0 1 2 3 4 5 6 7 8 9 10	-1 8 1 10 3 0 5 2 7 4 9 6
6 5 21 3 35 47 19 21	21 47 35 21 19 3