# CS109 2025 Spring Assignment 3

## 🥷 Rune Decryption (Normal, 30 points)

> Welcome to the Magic Academy of SUSTech! As newly recruited magic apprentices, you will learn how to decipher ancient runes, whose secrets are hidden in Java spells. Today, you will face 10 mysterious runes, and your task is to use your Java spells to decrypt them and earn the academy's rewards.

Each rune consists of a mixture of numbers and letters. Your goal is to write a decryption spell. The decryption rules are as follows:

1. **Splitting the Rune**: Separate the numeric and alphabetic parts of the rune.
2. **Processing Numbers**: For each numeric part, apply the following steps:
    1. If the **length of the number is odd**, remove the middle digit.
    2. Reverse the remaining digits.
    3. Convert the reversed digits into an integer.
3. **Processing Letters**: For each alphabetic part, apply the following steps:
    1. If the **length of the letter sequence is odd**, remove the middle letter.
    2. Convert each remaining letter into its corresponding ASCII code and sum all ASCII values.
4. **Sum Results**: Sum all processed numeric and alphabetic parts to obtain the final decrypted result.

**Notes**:

- Letter case matters, i.e., `a` -> `97`, `A` -> `65`.
- If the length of a numeric or alphabetic part is 1, removing the middle character results in an empty string, which should be treated as 0.

Grab your magic wand, tap the keyboard, and use Java spells to unlock these 10 mysterious runes and claim your academy rewards!

## Input Format

A single line containing a string representing a mysterious rune. The length `L` of the string is in the range `[1, 1000]`, and the rune consists only of numbers and letters.

## Output Format

A single line containing an integer, representing the decrypted numeric result. The numeric result will not exceed the range of an `int`.

## Sample Input

```
123abc9346a
```

## Sample Output

```
6666
```

## Explanation of the Sample

1. Split the rune into: numeric part `123`, alphabetic part `abc`, numeric part `9346`, and alphabetic part `a`.
2. Process numbers:
   - `123`: Length is odd, remove the middle `2`, resulting in `13`, then reverse it to `31`.
   - `9346`: Length is even, reverse it to `6439`.
3. Process letters:
   - `abc`: Length is odd, remove the middle `b`, resulting in `ac`. The ASCII code of `a` is `97`, and `c` is `99`. Their sum is `196`.
   - `a`: Length is odd, remove `a`, resulting in an empty string, which is not counted.
4. Sum results: `31 + 196 + 6439 = 6666`.

# □ Material Synthesis (Normal, 30 points)

> You are learning an ancient material synthesis magic. As apprentices of the Magic Academy of SUSTech, your task is to complete element synthesis according to specified magic rules, creating new magical materials and handing them over to the academy's magic research team for further study.

In this experiment, you will use two fundamental elements (Matrix A and Matrix B) and combine them through matrix multiplication with magical modifications to create a brand-new magical material (Matrix C). However, this synthesis is not a standard matrix multiplication; it is influenced by magical rules where certain additions are converted into subtractions, giving the material unique properties. Finally, you need to transpose the synthesized material so that the senior magicians of the academy can study it further. The synthesis rules are as follows:

1. Input two matrices `A` ($n \times m$) and `B` ($m \times p$) and perform matrix multiplication with subtraction to generate a new matrix `C` ($n \times p$):
    1. The calculation of `C` follows the standard matrix multiplication rule:
       $C_{ij} = A_{i1} \times B_{1j} + A_{i2} \times B_{2j} + \cdots + A_{im} \times B_{mj}$
    2. However, under the influence of magic, you need to invert the sign of every $k, 2k, 3k, 4k...$-th multiplication result to balance magical energy.
    3. For example, if `k=2` and `m=5`, then
       $C_{ij} = A_{i1} \times B_{1j} - A_{i2} \times B_{2j} + A_{i3} \times B_{3j} - A_{i4} \times B_{4j} + A_{i5} \times B_{5j}$
2. Transpose Matrix `C` to generate $C^T$, i.e., swapping rows and columns.

Grab your magic wand, tap the keyboard, and use Java spells to synthesize the most powerful magical material!

## Input Format

1. The first line contains four integers `n`, `m`, `p`, and `k`, representing:
    - Matrix `A` has a size of $n \times m$, meaning `n` rows and `m` columns.
    - Matrix `B` has a size of $m \times p$, meaning `m` rows and `p` columns.
    - `k` represents the interval number in the magic rule.
2. The next `n` lines, each containing `m` integers, represent Matrix `A`.
3. The next `m` lines, each containing `p` integers, represent Matrix `B`.

**Constraints**:

- All matrix elements are integers, and ($-1000 \leq A_{ij}, B_{ij} \leq 1000$).
- $1 \leq n, m, p \leq 100$
- $1 \leq k \leq 100$

## Output Format

Output the final synthesized matrix $C^T$, which has a size of $p \times n$, meaning `p` rows and `n` columns.

**Note**: The matrix output should be in row order, with elements separated by spaces. Please use nested loops to output the matrix, and **do not use** `Arrays.toString(matrix)` or `System.out.println(matrix)`.

## Sample Input

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

## Sample Output

```
10 31 52
8 26 44
```

## Explanation of the Sample

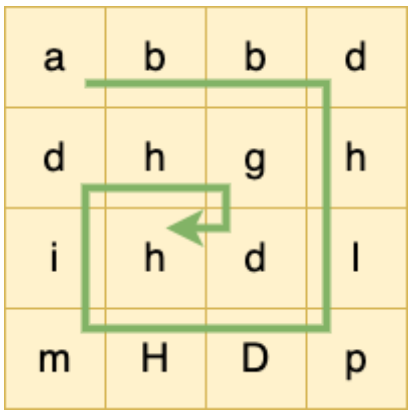
1. Compute matrix `C`:
    - $C_{11} = 1 \times 9 - 2 \times 7 + 3 \times 5 = 10$
    - $C_{12} = 1 \times 8 - 2 \times 6 + 3 \times 4 = 8$
    - $C_{21} = 4 \times 9 - 5 \times 7 + 6 \times 5 = 31$
    - $C_{22} = 4 \times 8 - 5 \times 6 + 6 \times 4 = 26$
    - $C_{31} = 7 \times 9 - 8 \times 7 + 9 \times 5 = 52$
    - $C_{32} = 7 \times 8 - 8 \times 6 + 9 \times 4 = 44$
2. Transpose matrix `C`

# 🛠 Basic Restoration (Hard, 40 points)

> At the Magic Academy of SUSTech, you are undergoing basic training in material restoration magic. Since you are still novice magic apprentices, this experiment only requires you to repair one type of damage, but you must completely restore all damaged areas to ensure the material is flawless!

Your goal is to find all the damaged runes in the magical material and replace them with the correct restoration runes, making the material whole again! The material is stored as an `n * m` character matrix, where each position contains a character. You need to traverse the entire matrix in a **clockwise spiral order** to find all damaged runes and replace them with the correct ones. The magic restoration rules are as follows:

1. **Read the Material**:
    1. Read an `n * m` character matrix `M`, representing the magical material that needs repair.
    2. Read two strings `old` and `new`:
        - `old` represents the damaged rune in the material.
        - `new` represents the correct restored rune.
        - `old` and `new` have the same length.
2. **Spiral Traversal of the Material**:
    1. Start from the top-left corner `(0,0)`, and traverse the entire matrix in a **clockwise spiral order**:
        - First, traverse from left to right along the first row.
        - Then, traverse from top to bottom along the last column.
        - Next, traverse from right to left along the last row.
        - Then, traverse from bottom to top along the first column.
        - Continue narrowing the range until the entire matrix is traversed.



2. Whenever `old` is found, replace it with `new`, then continue searching for the next occurrence until all matches are repaired.

3. **After restoration is complete, output the repaired material `M'`.**

**Notes**:

- `old` may not appear in `M`, or it may appear multiple times.
- If `old` is longer than the entire matrix, no replacement is needed.
- Letter case is **case-sensitive**, meaning `a` and `A` are considered different characters.
- Once replaced, a position will not be visited again, and it will not affect subsequent traversals.

Pick up your magic wand (keyboard), write your Java spell, and complete this **Basic Restoration Spell** challenge!

## Input Format

1. The first line contains two integers `n` and `m`, representing the size of the material matrix. That is, `n` rows and `m` columns.
2. The next `n` lines each contain `m` characters, representing the material matrix `M`.
3. The last two lines contain two strings `old` and `new`, representing the rune to be repaired and its restored version.

**Constraints**:

- $1 \leq n, m \leq 100$
- `old` and `new` have the same length, and their length does not exceed 10,000.
- The material matrix, `old`, and `new` contain only **uppercase and lowercase letters or digits**.

## Output Format

Output the repaired material matrix `M'`.

**Note**: The matrix output should be in row order, with elements separated by spaces. Please use nested loops to output the matrix, and **do not use** `Arrays.toString(matrix)` or `System.out.println(matrix)`.

## Sample Input 1

```
4 4
a b b d
d h g h
i h d l
m H D p
dh
xy
```

## Sample Output 1

```
a b b x
x y g y
i y x l
m H D p
```

## Explanation of Sample 1

The visit order is `a b b d h l p D H m i d h g d h`. Three occurrences of `dh` are found and replaced with `xy`. Since `DH` is different from `dh`, it is not replaced.

## Sample Input 2

```
3 3
a b c
d e f
g h i
ef
xy
```

## Sample Output 2

```
a b c
d e f
g h i
```

## Explanation of Sample 2

The visit order is `a b c f i h g d e`. `ef` does not appear in the matrix, so no replacement is needed.