INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CSC-101 PROGRAMMING WITH C AND C++

Autumn 2023 Problem Set 4

1. (Image Processing) Matrix transformations are fundamental in computer graphics and image processing. One common transformation is the combination of rotation and reflection.

   You are given an 'N x N' matrix containing only the characters '0' and '1'. You need to:

   1. Rotate the matrix 90 degrees clockwise.

   2. Reflect the rotated matrix along its main diagonal (from top-left to bottom-right).

   Your task is to write a program to perform these transformations and return the resulting matrix.

   Input:

   - An integer 'N' ($2 \leq N \leq 100$) representing the number of rows and columns of the matrix.

   - A matrix of size 'N x N' containing only characters '0' and '1'.

   Output:

   - A transformed matrix of size 'N x N'.

   Example:

   Input:

   N = 3

   Matrix:

```
1 1 0
0 0 1
1 0 0
```

   Output:

```
1 0 0
0 0 1
1 1 0
```

Explanation:

- After rotating 90 degrees clockwise:

```
1 0 1
0 0 1
0 1 0
```

- Reflecting the rotated matrix along the main diagonal:

```
1 0 0
0 0 1
1 1 0
```

Notes:

- It's beneficial to break the problem into two sub-problems: handling rotation and handling reflection.

- The problem does not require in-place transformation, but doing so can be an added challenge.

2. (Eigenvalue and Eigenvector Estimation) Given a 2x2 matrix A, write a program that calculates its eigenvalues and corresponding eigenvectors.

   Matrix A will be:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

   Inputs:

   - The elements of the matrix A (i.e., a, b, c, and d).

   Output:

   - Two eigenvalues.

   - The corresponding eigenvectors for each eigenvalue.

3. (The Magic Matrix) You're given a square matrix of integers (NxN) where N is an odd number. Your task is to write a program to:

   1. Determine the sum of the middle row.

   2. Replace every element of the main diagonal (from top-left to bottom-right) with this sum.

   3. For every element 'E' in the matrix, if 'E' is odd and not on the main diagonal, multiply it by the number of odd numbers in its row. If 'E' is even and not on the main diagonal, divide it by 2.

   4. Print the modified matrix.

   Input:

   The first line contains an integer 'N' ($3 \leq N \leq 11$), the dimension of the matrix. The next 'N' lines each contain 'N' integers, representing the matrix.

   Output:

   Print the modified matrix in the NxN format.

   Example:

   Input:

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

   Output:

```
15 1 3
15 15 9
1 9 15
```

4. (The Palindromic Shift) You are given an array of positive integers. Your task is to write a program that performs the following steps on each element of the array:

   1. Convert the integer into a string representation.

   2. If the string is a palindrome, divide the number by 2.

3. If the string is not a palindrome, shift the digits of the number cyclically to the right until it becomes a palindrome or until you've shifted as many times as the length of the number.

4. If the number becomes a palindrome after shifting, multiply it by the number of shifts required. If the number doesn't become a palindrome, add the sum of its digits to it.

Input:

The first line contains an integer 'N' ($1 \leq N \leq 100$), the size of the array. The next line contains 'N' positive integers, each not exceeding 10^6.

Output:

Print the modified array.

5. (The Caesar Cipher Array) A Caesar Cipher is one of the simplest encryption techniques where each letter in the plaintext is shifted a certain number of places down or up the alphabet. For example, with a shift of 1, 'A' would be replaced by 'B', 'B' would become 'C', and so on.

   However, in this challenge, instead of a single shift for the entire string, you will be given an array of shifts, and each character in your string will be shifted according to the corresponding value in the shift array.

   Your task is to write a program that takes a string and an array of integers (representing shifts) and returns the encrypted string. If the string is longer than the shift array, loop back to the beginning of the array for shifts. Only shift letters (both uppercase and lowercase); leave other characters unchanged.

   Input:

   - A string 'S' of length 'L' ($1 \leq L \leq 1000$) containing letters (both uppercase and lowercase) and possibly other characters.

   - An array of integers 'A' of size 'N' ($1 \leq N \leq 100$) where each integer is between 1 and 25 inclusive.

   Output:

   Print the encrypted string.

   Example:

   Input:

   String: "Hello, World!"

Shifts: [1, 2, 3, 4, 5]

Output:

“Igopt, Xqupi!”

Explanation:

For the given example:

1. ‘H’ is shifted by 1 to become ‘I’.
2. ‘e’ is shifted by 2 to become ‘g’.
3. ‘l’ is shifted by 3 to become ‘o’.
4. ‘l’ is shifted by 4 to become ‘p’.
5. ‘o’ is shifted by 5 to become ‘t’.
6. ‘,’ remains the same.
7. ‘ ’ (space) remains the same.
8. ‘W’ is shifted by 1 to become ‘X’.
9. ‘o’ is shifted by 2 to become ‘q’.
10. ‘r’ is shifted by 3 to become ‘u’.
11. ‘l’ is shifted by 4 to become ‘p’.
12. ‘d’ is shifted by 5 to become ‘i’.
13. ‘!’ remains the same.

6. (The Artistic Spiral) An artist wants to draw spirals but with a twist! Instead of a typical spiral that has a uniform curve, they want a spiral that curves based on an input sequence of numbers.

   Imagine you’re on a 2D plane, starting at ‘(0, 0)’. Given a sequence of numbers, for each number, ‘n’, you will:

   1. Draw a line straight up for ‘n’ units.
   2. Turn 90 degrees to the right.
   3. Draw a line for ‘n’ units.
   4. Turn 90 degrees to the right again, and so on . . .

   Your task is to find the final coordinates after drawing the entire sequence.

   Input:

- A list of integers ‘L’ of size ‘N’ ($1 \leq N \leq 1000$) where each integer is between 1 and 1000 inclusive.

Output:

Print the final ‘(x, y)’ coordinates after drawing the entire sequence.

Example:

Input:

Sequence: [3, 2, -1, 4]

Output:

(7, 3)

Explanation:

Starting from ‘(0,0)’,

1. Go up 3 units to ‘(0,3)’.
2. Turn right and go 2 units to ‘(2,3)’.
3. Turn left and go 1 unit down to ‘(3,3)’.
4. Turn right and go 4 units to ‘(7,3)’.

7. (The Bookshelf Balancer) You have been tasked with organizing a bookshelf. The bookshelf is peculiar: it prefers symmetry. Each book has a certain weight, and the shelf wants the books to be placed in such a way that the cumulative weight from the left side to the center is as close as possible to the cumulative weight from the center to the right side. If the number of books is even, consider the two centermost books as the center.

   However, there's a catch! You can only swap books. No removals or rearrangements allowed except through swapping.

   Write a program to determine the minimum number of swaps required to achieve the best balance possible. If perfect balance is not possible, get as close as you can. Return the number of swaps required and the difference in weight between the two halves.

   Input:

   - A list of integers ‘W’ representing the weight of each book in the order they are currently placed on the shelf. The size of the list ‘N’ ($2 \leq N \leq 100$) where each integer weight is between 1 and 1000 inclusive.

Output:

- An integer representing the minimum number of swaps required.

- An integer representing the weight difference between the two halves after the swaps.

8. (Reflections in 2D Plane) In a two-dimensional plane, there is a point P and a line L. When light is emitted from point P and strikes the line L, it reflects off the line according to the law of reflection: the angle of incidence is equal to the angle of reflection.

   Given the coordinates of point P, the equation of the line L in the form $ax+by+c=0$, and a distance d, find the coordinates of point Q such that:

   1. Point Q lies on the ray of light reflected off the line L.

   2. The distance between P and the line L is the same as the distance between Q and the line L.

   3. The distance from the point of reflection R on line L to point Q is d.

   Input:

   1. $P(x_1, y_1)$ - Coordinates of the point P.

   2. a, b, c - The coefficients of the equation of line L.

   3. d - A distance.

   Output:

   Coordinates $Q(x_2, y_2)$.

   Example:

   Input:

   Point P: (1, 2)

   Line L: x + y - 5 = 0

   Distance d: $2\sqrt{2}$

   Output:

   Point Q: (3, 4)

9. (Character Pattern Analysis) In linguistics, it's known that certain characters often appear together in a sequence more frequently than others. Recognizing these patterns can be essential for tasks like compression, cryptography, or even predicting the next character in a sequence.

   You are given a long text string and are asked to find the most frequent 2-character sequence (bigram) in that text. If multiple bigrams have the same highest frequency, return all of them. The text will only contain lowercase English alphabets and spaces, and words will be separated by a single space.

   Write a program to:

   1. Parse the input string.

   2. Identify and count bigrams.

   3. Return the most frequent bigram(s).

   Input:

   - A string 's' (length: $1 \leq |s| \leq 10^5$) containing lowercase English alphabets and spaces.

   Output:

   - A list of most frequent 2-character bigram(s).

   Example:

   Input:

   "it was the best of times it was the worst of times"

   Output:

   ["it", "th", "wa", "he", "as"]

   Explanation:

   The bigrams "it", "th", "wa" etc. all appear three times in the string, which is more than any other bigrams.

10. (String Sequence Detector) A repetitive sequence in a string refers to the repetition of a set of characters. Sometimes, these sequences are nested within each other, creating patterns.

   You're given a string 'S' of length 'N' ($1 \leq \text{N} \leq 1000$). Your task is to detect:

   1. The shortest repeating sequence.

2. The number of times the sequence repeats consecutively without any other character interrupting the pattern.

If there are multiple sequences with the same length, return the one which occurs first in the string.

Input:

- A string 'S' containing only lowercase English letters.

Output:

- The shortest repeating sequence and its consecutive repetition count. If no such sequence exists, return "No repeating sequence found".

Example:

Input:

S = "abcabcabcxyzxyzaaa"

Output:

Sequence: abc, Count: 3

Explanation:

The string "abcabcabcxyzxyzaaa" has a repetitive sequence "abc" that repeats consecutively 3 times. There's another sequence "xyz" that repeats twice, but "abc" is the one that occurs first and is returned.

Notes:

- Start by identifying patterns in the string.

- Nested patterns can be a challenge. For instance, in the string "abcabcabcab", although "abcab" repeats twice, the actual repeating sequence is "abc" since "abcab" isn't a complete repetition.