IMAGE FLIP

```
class Solution {
    public int[][] flipAndInvertImage(int[][] image) {
        int[][] result = new int[image.length][image.length]; //new array for the result
        for(int i = 0; i<image.length; i++){
            for(int j = 0; j<image.length; j++){
                result[i][j] = image[i][image.length - 1 - j]; //flipping horizontally
                if(result[i][j] == 0){ // inverting the matrix
                    result[i][j] = 1;
                }else if(result[i][j] == 1){
                    result[i][j] = 0;
                }
            }
        }
        return result;

    }
}
```

MAT MUL

```
class Solution {
  public int[][] multiply(int[][] mat1, int[][] mat2) {
    final int m = mat1.length;
    final int n = mat2.length;
    final int l = mat2[0].length;
    int[][] ans = new int[m][l];

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < l; ++j)
        for (int k = 0; k < n; ++k)
          ans[i][j] += mat1[i][k] * mat2[k][j];

    return ans;
  }
}
```

IS PREFIX == SUFFIX

```
import java.util.Scanner;

public class PrefixSuffixCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input the strings
        String[] input = scanner.nextLine().split(" ");
        String S = input[0];
        String T = input[1];

        // Check if T is both a prefix and a suffix of S
        boolean isPrefix = true;
        boolean isSuffix = true;

        int lenS = S.length();
```

```
        int lenT = T.length();

        for (int i = 0; i < lenT; i++) {
            if (S.charAt(i) != T.charAt(i)) {
                isPrefix = false;
                break;
            }
        }

        for (int i = 0; i < lenT; i++) {
            if (S.charAt(lenS - lenT + i) != T.charAt(i)) {
                isSuffix = false;
                break;
            }
        }

        if (isPrefix && isSuffix) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }

        scanner.close();
    }
}
```

LOGEST PREFIX SUFFIX

```
import java.util.Scanner;

public class LongestPrefixSuffix {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input the string
        String S = scanner.next();

        int len = computeLongestPrefixSuffix(S);
        System.out.println(len);

        scanner.close();
    }

    public static int computeLongestPrefixSuffix(String str) {
        int n = str.length();
        int[] lps = new int[n];
        int len = 0; // Length of the previous longest prefix suffix

        lps[0] = 0; // lps[0] is always 0

        int i = 1;
        while (i < n) {
            if (str.charAt(i) == str.charAt(len)) {
                len++;
                lps[i] = len;
```

```
            i++;
          } else {
            if (len != 0) {
              len = lps[len - 1];
            } else {
              lps[i] = 0;
              i++;
            }
          }
        }

        // The length of the longest proper prefix which is also a suffix
        return lps[n - 1];
    }
}
```

SPARSE MAT

```
public class Solution {
    /**
     * @param matrix: a matrix of integers
     * @return: an array of integers
     */
    public int[] printZMatrix(int[][] matrix) {
        int[] rst = null;
        if (matrix == null || matrix.length == 0 || matrix[0] == null || matrix[0].length == 0) {
            return rst;
        }
        int n = matrix.length;
        int m = matrix[0].length;
        rst = new int[n * m];
        if (matrix.length == 1) {
            return matrix[0];
        }
        int i = 0, j = 0;
        int ind = 0;
        rst[ind] = matrix[i][j];
        ind++;
        while (ind < rst.length) {
            //Right 1 step, or down
                    if (j + 1 < m || i + 1 < n) {
                            if (j + 1 < m) j++;
                            else if (i + 1 < n) i++;
                            rst[ind++] = matrix[i][j];
                    }
                    //Move left-bottom:
                    while (j - 1 >= 0 && i + 1 < n) {
                                    rst[ind++] = matrix[++i][--j];
                    }
            //Move down, or right
                    if (j + 1 < m || i + 1 < n) {
                            if (i + 1 < n) i++;
                            else if (j + 1 < m) j++;
                            rst[ind++] = matrix[i][j];
                    }
            //Move right-up:
                    while (j + 1 < m && i - 1 >= 0) {
```

```
                        rst[ind++] = matrix[--i][++j];
                    }
        }//end while
        return rst;
    }
}

```