DATE : 18.12.2020
SESSION : AFTERNOON

Rohinidevi S V
312217104131
CSE - C

IT8761 - SECURITY LABORATORY

Q. Develop a java program to find the inverse of the given matrix and decrypt the message using Hill Cipher. (Key will be a numerical matrix)

AIM:

To find the inverse of the given matrix and decrypt the message using Hill Cipher.

PROCEDURE :

- **Inverse of the key matrix**
  1. Get the key matrix as input.
  2. Compute the determinant of the key matrix.
  3. If the determinant is zero, then an inverse for the matrix doesnot exist. Hence decryption cannot be done with this key matrix. Stop the procedure.
  4. Otherwise, compute the inverse of the determinant and thereby find the inverse matrix.
- **Decryption Procedure for Hill Cipher.**
  1. Use the cipher text as input.
  2. Split them into tri-graph.
  3. Convert each letter into its equivalent numerical representation.
  4. Obtain the columnar matrix.
  5. To retrieve the plaintext from the cipher text,

use the following formula,

$$P = K^{-1}C \bmod 26$$

6. Convert the product matrix back into letter representation.
7. This the required plaintext.
8. Display the plaintext.

SAMPLE INPUT AND OUTPUT :

Enter Key Matrix :

1 0 2

10 20 15

0 1 2

INVERSE MATRIX :

15 22 2

14 22 3

6 15 12

Enter Cipher Text :

lqwoufwgwojya

CIPHER TEXT MATRIX :

11 20 6 9

16 5 22 24

22 22 14 0

PLAINTEXT :

paymoremoney

SOURCE CODE

```
// to find the inverse of a given matrix
// to decrypt the ciphertext using Hill Cipher

import java.util.*;

class Main {
  public static void displayMatrix(int[][] mat) {
    for(int i = 0; i < mat.length; i++) {
      for(int j = 0; j < mat[i].length; j++) {
        System.out.print(mat[i][j] + " ");
      }
      System.out.println();
    }
  }

  public static boolean isProperText(String text) {
    return text.matches("^[a-zA-Z]*$");
  }

  public static String generateText(int[][] matrix) {
    StringBuilder text = new StringBuilder();
    for(int j = 0; j < matrix[0].length; j++) {
      for(int i = 0; i < matrix.length; i++) {
        text.append((char) (matrix[i][j] + 97));
      }
    }
    return text.toString();
  }

  public static int[][] getCofactor(int[][] A, int p, int q, int n) {
    int[][] temp = new int[3][3];
    int i = 0, j = 0;
    for(int row = 0; row < n; row++) {
      for(int col = 0; col < n; col++) {
        if(row != p && col != q) {
          temp[i][j++] = A[row][col];
          if(j == n - 1) {
            j = 0;
            i++;
          }
        }
      }
    }
    return temp;
  }
```

```
public static int findDeterminant(int[][] matrix, int n) {
  int D = 0;
  if(n == 1) return matrix[0][0];
  int sign = 1;
  for(int f = 0; f < n; f++) {
    int[][] temp = getCofactor(matrix, 0, f, n);
    D += sign * matrix[0][f] * findDeterminant(temp, n - 1);
    sign = -sign;
  }
  return D;
}

public static int findMultiplicativeInverse(int det, int m) {
  int inv;
  for(inv = 0; inv < 26; inv++) {
    if((det * inv) % m == 1) {
      break;
    }
  }
  if(inv == 26) return -1;
  return inv;
}

public static int[][] findAdjoint(int[][] A, int n) {
  int[][] adj = new int[3][3];
  if(n == 1) {
    adj[0][0] = 1;
    return adj;
  }
  int sign = 1;
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
      int[][] temp = getCofactor(A, i, j, n);
      sign = ((i + j) % 2 == 0) ? 1 : -1;
      adj[j][i] = (sign) * (findDeterminant(temp, n - 1));
    }
  }
  return adj;
}

public static int[][] findInverseMatrix(int[][] matrix) {
  int[][] inverse = new int[3][3];
  int det = findDeterminant(matrix, 3);
  if(det == 0) {
    System.out.println("\nWARNING:");
    System.out.println("Inverse does not exist.");
    return null;
  }
```

```
  int det_inv = findMultiplicativeInverse(det, 26);
  if(det_inv == -1) {
    System.out.println("\nWARNING:");
    System.out.println("Determinant is " + det + ".");
    System.out.println("Modulo Multiplicative Inverse does not exist.");
    return null;
  }
  int[][] adj = findAdjoint(matrix, 3);
  for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 3; j++) {
      int val = (det_inv * adj[i][j]) % 26;
      if(val < 0) {
        inverse[i][j] = 26 + val;
      } else {
        inverse[i][j] = val;
      }
    }
  }
  return inverse;
}

public static int[][] createMatrix(String text, int rows, int cols) {
  int[][] matrix = new int[rows][cols];
  int pos = 0;
  for(int j = 0; j < cols; j++) {
    for(int i = 0; i < rows; i++) {
      matrix[i][j] = (int) (text.charAt(pos++) - 97);
    }
  }
  return matrix;
}

public static int[][] multiplyMatrices(int[][] a, int[][] b) {
  int rows = a.length;
  int cols = b[0].length;
  int[][] matrix = new int[rows][cols];
  for(int i = 0; i < rows; i++) {
    for(int j = 0; j < cols; j++) {
      matrix[i][j] = 0;
      for(int k = 0; k < a[i].length; k++) {
        matrix[i][j] += a[i][k] * b[k][j];
      }
      matrix[i][j] %= 26;
    }
  }
  return matrix;
}
```

```
public static String encrypt(String plain_text, int[][] key_mat) {
  int[][] text_mat = createMatrix(plain_text, 3, plain_text.length() / 3);
  System.out.println("\nPLAIN TEXT MATRIX:");
  displayMatrix(text_mat);
  int[][] product_mat = multiplyMatrices(key_mat, text_mat);
  return generateText(product_mat);
}

public static String decrypt(String cipher_text, int[][] inv_mat) {
  int[][] text_mat = createMatrix(cipher_text, 3, cipher_text.length() / 3);
  System.out.println("\nCIPHER TEXT MATRIX:");
  displayMatrix(text_mat);
  int[][] product_mat = multiplyMatrices(inv_mat, text_mat);
  return generateText(product_mat);
}

public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
  System.out.println("--- HILL CIPHER ---");
  int[][] key_mat = new int[3][3];
  System.out.println("Enter the Key Matrix:");
  for(int i = 0; i < 3; i++) {
    for(int j = 0; j < 3; j++) {
      key_mat[i][j] = sc.nextInt();
    }
  }
  int[][] inv_mat = findInverseMatrix(key_mat);
  if(inv_mat != null) {
    System.out.println("INVERSE MATRIX:");
    displayMatrix(inv_mat);
    System.out.println("\n- DECRYPTION");
    int flag = 0;
    String plain_text = "", cipher_text = "";
    do {
      if(flag == 1) {
        System.out.println("\nWARNING:");
        System.out.println("Plain Text must contain only alphabets");
      }
      sc.nextLine();
      System.out.println("\nEnter Plain Text:");
      plain_text = sc.nextLine();
      flag = 1;
    }while(!isProperText(plain_text));
    if(plain_text.length() % 3 != 0) {
      while(plain_text.length() % 3 != 0) {
        plain_text += "x";
      }
    }
```

```
      String enc_text = encrypt(plain_text, key_mat);
      System.out.println("\nCIPHER TEXT:\n" + enc_text);
      flag = 0;
      do {
        if(flag == 1) {
          System.out.println("\nWARNING:");
          System.out.println("Cipher Text must contain only alphabets");
        }
        System.out.println("\nEnter Cipher Text:");
        cipher_text = sc.nextLine();
        flag = 1;
      }while(!isProperText(cipher_text));
      String dec_text = decrypt(cipher_text, inv_mat);
      System.out.println("PLAIN TEXT:\n" + dec_text);
    }
    sc.close();
  }
}
```

OUTPUT

```
--- HILL CIPHER ---
Enter the Key Matrix:
1 0 2
10 20 15
0 1 2
INVERSE MATRIX:
15 22 2
14 22 3
6 15 12

- DECRYPTION

Enter Plain Text:
paymoremoney



PLAIN TEXT MATRIX:
15 12 4 13
0 14 12 4
24 17 14 24

CIPHER TEXT:
lqwufwgwojya

Enter Cipher Text:
lqwufwgwojya

CIPHER TEXT MATRIX:
11 20 6 9
16 5 22 24
22 22 14 0
PLAIN TEXT:
paymoremoney
```