**Name - Gaurang A Raorane Roll No - 49**

**Class - D15A Batch - C**

**EXPERIMENT 2**

**Aim:-** Design and Implement a product cipher using Substitution ciphers and Transposition Cipher.

import java.util.Scanner;

public class product\_cipher {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the plaintext: ");

String plaintext = scanner.nextLine();

System.out.print("Enter the transposition key (a number): ");

int transpositionKey = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter the substitution key (a string of unique characters): ");

String substitutionKey = scanner.nextLine();

String encryptedText = encrypt(plaintext, transpositionKey, substitutionKey);

System.out.println("Encrypted text: " + encryptedText);

String decryptedText = decrypt(encryptedText, transpositionKey, substitutionKey);

System.out.println("Decrypted text: " + decryptedText);

}

public static String encrypt(String plaintext, int transpositionKey, String substitutionKey) {

String substitutionResult = substitute(plaintext, substitutionKey);

String transpositionResult = transpose(substitutionResult, transpositionKey);

return transpositionResult;

}

public static String decrypt(String ciphertext, int transpositionKey, String substitutionKey) {

String reverseTranspositionResult = reverseTranspose(ciphertext, transpositionKey);

String reverseSubstitutionResult = reverseSubstitute(reverseTranspositionResult, substitutionKey);

return reverseSubstitutionResult;

}

public static String substitute(String text, String key) {

StringBuilder result = new StringBuilder();

for (char c : text.toCharArray()) {

int index = key.indexOf(c);

if (index != -1) {

result.append(key.charAt((index + 1) % key.length())); // Simple substitution logic

} else {

result.append(c);

}

}

return result.toString();

}

public static String reverseSubstitute(String text, String key) {

StringBuilder result = new StringBuilder();

for (char c : text.toCharArray()) {

int index = key.indexOf(c);

if (index != -1) {

int newIndex = (index - 1 + key.length()) % key.length();

result.append(key.charAt(newIndex));

} else {

result.append(c);

}

}

return result.toString();

}

public static String transpose(String text, int key) {

int numRows = (int) Math.ceil((double) text.length() / key);

char[][] matrix = new char[numRows][key];

int index = 0;

for (int i = 0; i < numRows; i++) {

for (int j = 0; j < key && index < text.length(); j++) {

matrix[i][j] = text.charAt(index++);

}

}

StringBuilder result = new StringBuilder();

for (int j = 0; j < key; j++) {

for (int i = 0; i < numRows; i++) {

if (matrix[i][j] != '\0') {

result.append(matrix[i][j]);

}

}

}

return result.toString();

}

public static String reverseTranspose(String text, int key) {

int numRows = (int) Math.ceil((double) text.length() / key);

char[][] matrix = new char[numRows][key];

int index = 0;

for (int j = 0; j < key; j++) {

for (int i = 0; i < numRows && index < text.length(); i++) {

matrix[i][j] = text.charAt(index++);

}

}

StringBuilder result = new StringBuilder();

for (int i = 0; i < numRows; i++) {

for (int j = 0; j < key; j++) {

if (matrix[i][j] != '\0') {

result.append(matrix[i][j]);

}

}

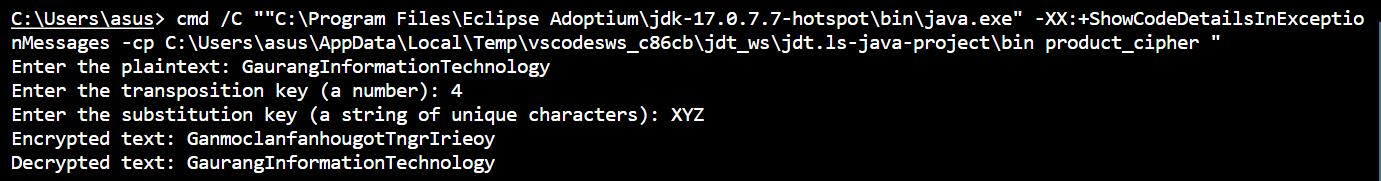
}

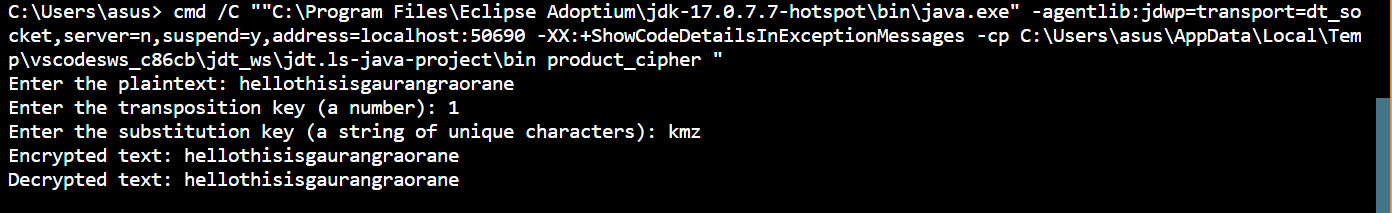
return result.toString();

}

}

Output:-



****