University of Engineering & Management
Institute of Engineering & Management

श्रद्धाबान लभते ज्ञानम्
Good Education, Good Jobs

Institute of Engineering & Management, University of Engineering & Management Kolkata

Department of Computer Science & Engineering

Laboratory: - Introduction to Cyber Security Lab

Name: - Gourab Mondal

Section: - B

Roll Number: - 26

Enrollment Number: - 12022002001116

Year: - 2022-26

INDEX PAGE

Week 1

1a.

Title: Implementation of Caesar Cipher encryption and decryption using java.

Algorithm:

1. Take the input string (text) and a shift value (key).
2. For each character in the string:
    - If it's a letter:
        - Shift it by key positions in the alphabet (wrap around if needed).
    - Leave non-alphabetic characters unchanged.
3. Return the modified string as the encrypted or decrypted text, based on the shift direction (positive for encryption, negative for decryption).

Java Program:

```
class CaesarCypher {
    private int keyVal;

    CaesarCypher() {
        this(3);
    }

    CaesarCypher(int keyVal) {
        if (keyVal > 0) {
            this.keyVal = keyVal;
        }
    }

    private String formatText(String text) {
        text = text.toLowerCase();
        return text.replaceAll("[^a-z ]", "");
    }

    private char shiftWithWraparound(char c, int key) {
        if (c == ' ')
            return ' ';
```

```
        char base = 'a';
        int shiftedChar = (c - base + key) % 26;


        if (shiftedChar < 0) {
            shiftedChar += 26;
        }

        return (char) (shiftedChar + base);
    }

    public String encrypt(String text) {
        String formattedText = this.formatText(text);
        StringBuilder sb = new StringBuilder();
        for (char c : formattedText.toCharArray()) {
            sb.append(this.shiftWithWraparound(c, this.keyVal));
        }

        return sb.toString();
    }

    public String decrypt(String cipher) {
        String formattedText = this.formatText(cipher);
        StringBuilder sb = new StringBuilder();
        for (char c : formattedText.toCharArray()) {
            sb.append(this.shiftWithWraparound(c, -this.keyVal));
        }

        return sb.toString();
    }
}
```

Output:

```
PS C:\Users\GOURAB\OneDrive\Desktop\all_sub> cd "c:\Users\GOURAB\
$?) { javac CaesarCypher.java } ; if ($?) { java CaesarCypher }
Original: xyz abc, Encrypted: abc def, Decrypted: xyz abc
```

1b.

Title: Implementation of Playfair Cipher encryption and decryption using java.

Algorithm:

1. **Create a 5x5 key matrix** using a keyword (remove duplicates, exclude 'J').
2. **Prepare the plaintext**: Replace 'J' with 'I', split into digraphs, add filler ('X') for duplicates/odd length.
3. **Encrypt digraphs**:
    - Same row: Shift right.
    - Same column: Shift down.
    - Rectangle: Swap columns.
4. **Cipher Text**: Combine the result to form the ciphertext.
5. **Decrypt digraphs**: reverse the shifts (left for rows, up for columns) and remove fillers.

Java Program:

```
public class PlayfairCypher {
    private static final String DEFAULT_KEY = "Monarchy";
    private final int MAT_SIZE = 5;

    private String key;
    private char[][] keyMat = new char[MAT_SIZE][MAT_SIZE];
    private LinkedHashSet<Character> keySet = new LinkedHashSet<>();
    private List<Character> alphabets = new ArrayList<>();

    public PlayfairCypher() {
        this(DEFAULT_KEY);
    }

    public PlayfairCypher(String key) {
        if (key != null && key.length() != 0) {
            this.key = key;
        } else {
            this.key = DEFAULT_KEY;
        }
```

```
        this.populateAlphabets();
        String formattedKey = this.formatKey(this.key);
        this.createKeyMatrix(formattedKey);
    }

    private void populateAlphabets() {
        for (char c = 'a'; c <= 'z'; c++) {
            if (c != 'j') {
                this.alphabets.add(c);
            }
        }
    }

    private String formatKey(String key) {
        key = key.toLowerCase();
        key = key.replace('j', 'i');
        key = key.replaceAll("[^a-zA-Z]", "");
        try {
            return this.stripDuplicateChars(key);
        } catch (IllegalArgumentException e) {

            System.out.println("Error: " + e.getMessage());
            return "";
        }
    }

    private String stripDuplicateChars(String str) {
        if (str == null || str.length() == 0) {
            throw new IllegalArgumentException("Input string cannot be null or
empty");
        }

        for (char c : str.toCharArray()) {
            this.keySet.add(c);
        }

        StringBuilder sb = new StringBuilder();
        for (char c : this.keySet) {
            sb.append(c);
        }

        return sb.toString();
    }
```

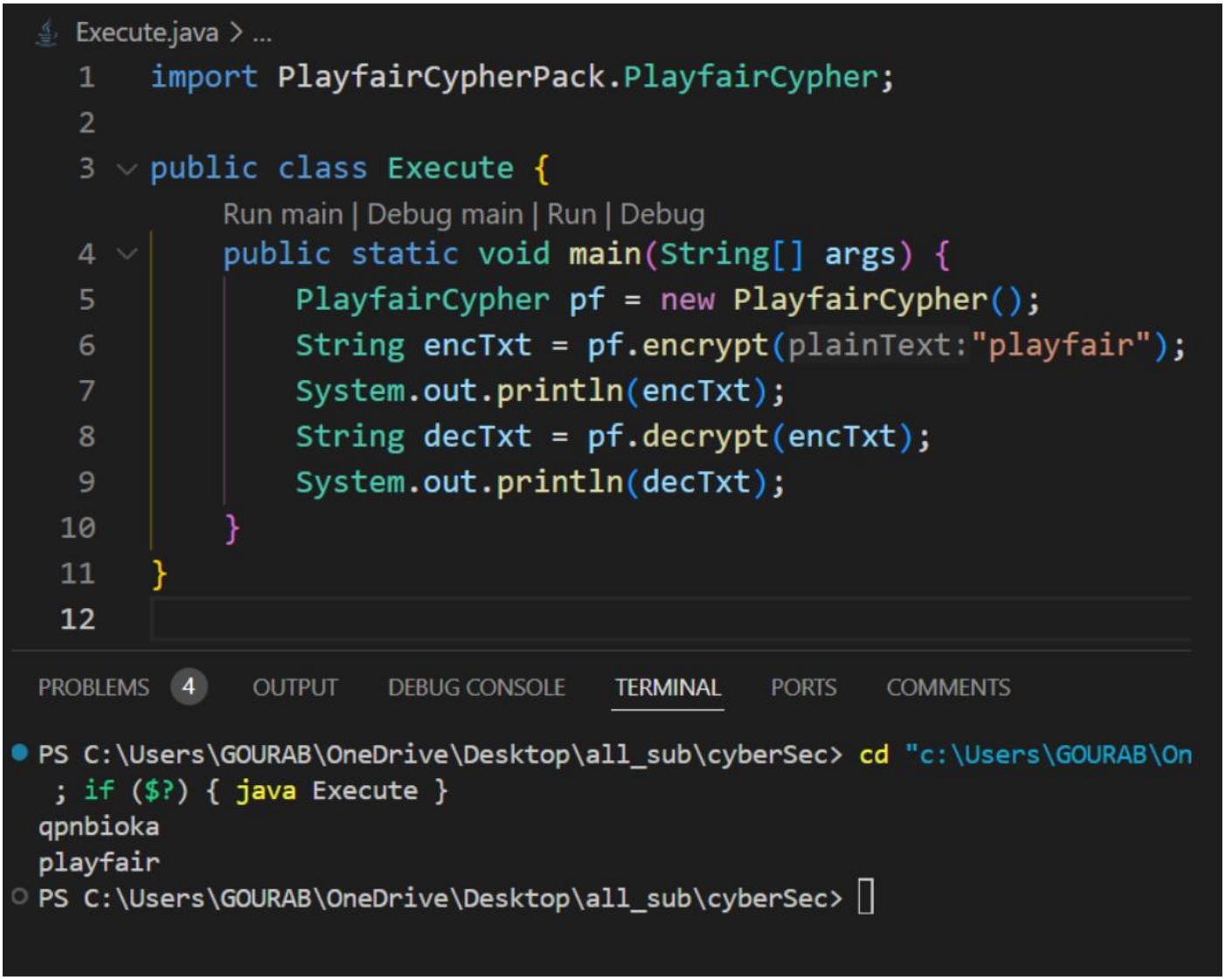
```
    private void createKeyMatrix(String key) {
        int s = 0;
        int k = 0;
        for (int i = 0; i < this.MAT_SIZE; i++) {
            for (int j = 0; j < this.MAT_SIZE; j++) {
                if (s < key.length()) {
                    this.keyMat[i][j] = key.charAt(s++);
                } else {

                    while (this.keySet.contains(this.alphabets.get(k))) {
                        k++;
                    }
                    this.keyMat[i][j] = this.alphabets.get(k++);
                }
            }
        }
    }

    private List<String> createDigraphs(String text) {
        if (text == null || text.length() == 0) {
            throw new IllegalArgumentException("Input string cannot be null or
empty");
        }
        ArrayList<String> digraphs = new ArrayList<>();
        ArrayList<Character> charList = new ArrayList<>();
        for (char c : text.toCharArray()) {
            charList.add(c);
        }

        for (int i = 0; i + 1 < charList.size(); i += 2) { // if duplicate pair
is found x is appended and all items
                                                           // shifted
            if (charList.get(i) == charList.get(i + 1)) {
                charList.add(i + 1, 'x');
            }
        }
        if (charList.size() % 2 != 0) { // if odd length of char list it adds
one extra x at the end
            charList.add('x');
        }
```

```
        for (int i = 0; i + 1 < charList.size(); i += 2) { // takes pairs from
the char list and adds as string into
                                                           // digraphs
            digraphs.add(String.valueOf(charList.get(i)) +
String.valueOf(charList.get(i + 1)));
        }

        return digraphs;
    }

    private int[] getIndex(char c) {
        int i, j;
        for (i = 0; i < this.keyMat.length; i++) {
            for (j = 0; j < this.keyMat.length; j++) {
                if (this.keyMat[i][j] == c) {
                    return new int[] { i, j };
                }
            }
        }
        return null;
    }

    private String getCypher(String digraph) {
        int[] first = this.getIndex(digraph.charAt(0)); // Get indices of first
char
        int[] second = this.getIndex(digraph.charAt(1)); // Get indices of
second char

        int i1 = first[0], j1 = first[1];
        int i2 = second[0], j2 = second[1];

        char char1, char2;

        if (i1 == i2) { // Same row
            char1 = this.keyMat[i1][(j1 + 1) % MAT_SIZE]; // Shift right, wrap
around
            char2 = this.keyMat[i2][(j2 + 1) % MAT_SIZE];
        } else if (j1 == j2) { // Same column
            char1 = this.keyMat[(i1 + 1) % MAT_SIZE][j1]; // Shift down, wrap
around
            char2 = this.keyMat[(i2 + 1) % MAT_SIZE][j2];
        } else { // Different row and column
```

```
            char1 = this.keyMat[i1][j2]; // Swap columns
            char2 = this.keyMat[i2][j1];
        }

        return String.valueOf(char1) + String.valueOf(char2);
    }

    private String getDecypher(String digraph) {
        int[] first = this.getIndex(digraph.charAt(0)); // Get indices of first
char
        int[] second = this.getIndex(digraph.charAt(1)); // Get indices of
second char

        int i1 = first[0], j1 = first[1];
        int i2 = second[0], j2 = second[1];

        char char1, char2;

        if (i1 == i2) { // Same row
            char1 = this.keyMat[i1][(j1 - 1 + MAT_SIZE) % MAT_SIZE]; // Shift
left, wrap around
            char2 = this.keyMat[i2][(j2 - 1 + MAT_SIZE) % MAT_SIZE];
        } else if (j1 == j2) { // Same column
            char1 = this.keyMat[(i1 - 1 + MAT_SIZE) % MAT_SIZE][j1]; // Shift
up, wrap around
            char2 = this.keyMat[(i2 - 1 + MAT_SIZE) % MAT_SIZE][j2];
        } else { // Different row and column
            char1 = this.keyMat[i1][j2]; // Swap columns
            char2 = this.keyMat[i2][j1];
        }

        return String.valueOf(char1) + String.valueOf(char2);
    }

    public String encrypt(String plainText) {
        List<String> digraphs = this.createDigraphs(plainText);
        List<String> cipherText = new ArrayList<>();

        for (String digraph : digraphs) {
            cipherText.add(this.getCypher(digraph));
        }
```

```
        return String.join("", cipherText);
    }

    public String decrypt(String cipherText) {
        List<String> digraphs = this.createDigraphs(cipherText); // Create
digraphs from ciphertext
        List<String> plainText = new ArrayList<>();

        for (String digraph : digraphs) {
            plainText.add(this.getDecypher(digraph)); // Get decrypted digraph
        }

        String decryptedText = String.join("", plainText);

        // Remove filler 'x' characters (assuming 'x' was used during
encryption)
        decryptedText = decryptedText.replaceAll("x$", ""); // Remove trailing
'x' (optional)
        decryptedText = decryptedText.replaceAll("x", ""); // Remove all filler
'x'

        return decryptedText;
    }

    public void displayKeyMatrix() {
        for (int i = 0; i < MAT_SIZE; i++) {
            for (int j = 0; j < MAT_SIZE; j++) {
                System.out.print(keyMat[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output:


Execute.java > ...
1 import PlayfairCypherPack.PlayfairCypher;
2
3 public class Execute {
Run main | Debug main | Run | Debug
4 public static void main(String[] args) {
5 PlayfairCypher pf = new PlayfairCypher();
6 String encTxt = pf.encrypt(plainText:"playfair");
7 System.out.println(encTxt);
8 String decTxt = pf.decrypt(encTxt);
9 System.out.println(decTxt);
10 }
11 }
12
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
PS C:\Users\GOURAB\OneDrive\Desktop\all_sub\cyberSec> cd "c:\Users\GOURAB\On
; if ($?) { java Execute }
qpnbioka
playfair
PS C:\Users\GOURAB\OneDrive\Desktop\all_sub\cyberSec>