Somaiya Vidyavihar University
K J Somaiya School of Engineering

Course Name:	**Information Security (116U01L602)**	**Semester:**	VI
Date of Performance:	22 / 01 / 2025	**DIV/ Batch No:**	A-4
Student Name:	Hyder Presswala	**Roll No:**	16010122151

Title: Encryption-Decryption programs using classical cryptography (Playfair cipher, Transposition cipher)

Objectives:

To write a program to convert plain text into cipher text using Caesar cipher and Transposition cipher

Expected Outcome of Experiment:

CO1 :- Explain various security goals, threats, vulnerabilities and controls
CO2 :- Apply various cryptographic algorithms for software security

Books/ Journals/ Websites referred:

1. Security in Computing
2. Cryptography and Network Security
3. Cryptography and Network Security: Principles and Practice

New Concepts to be learned:

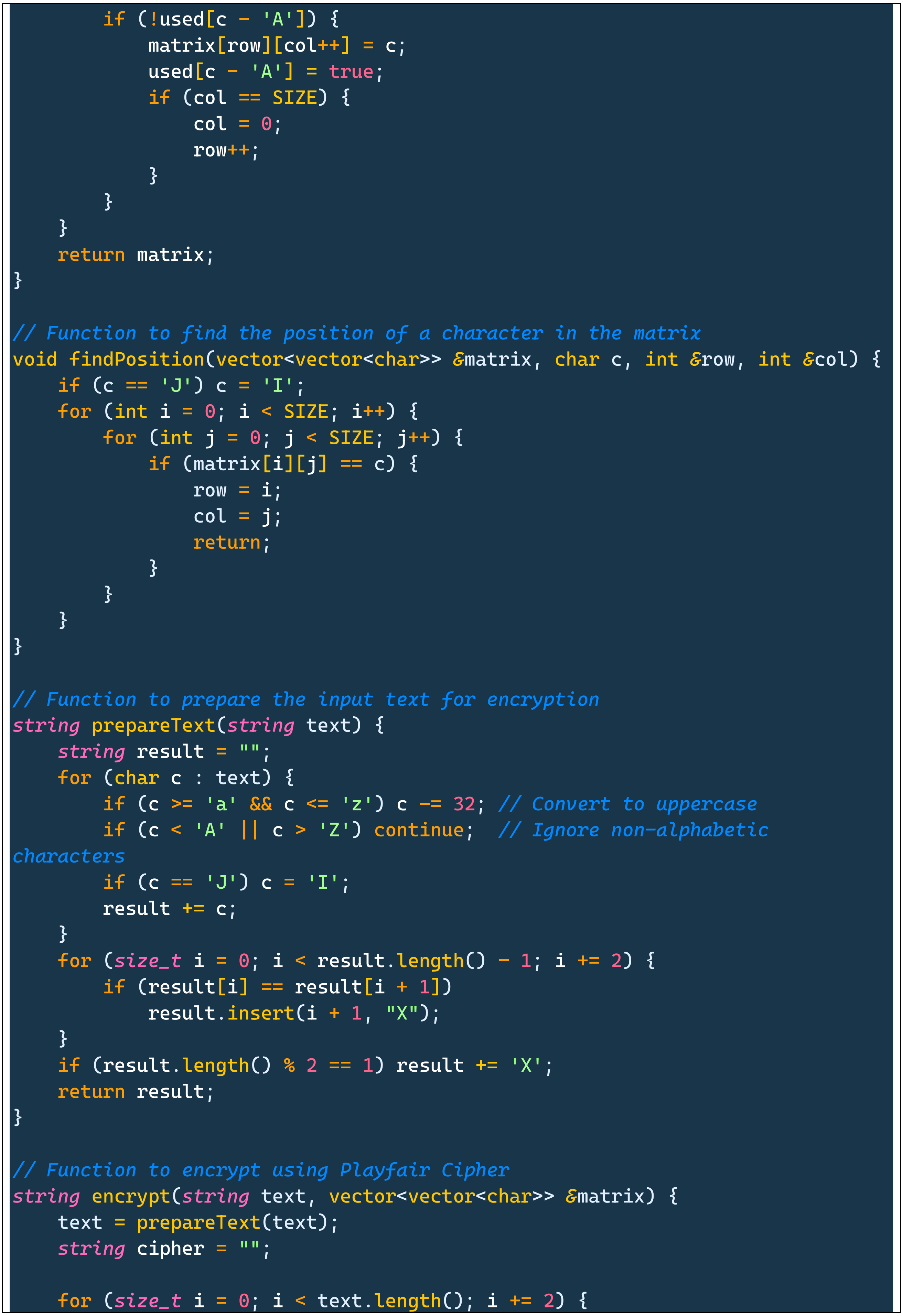
Encryption and Decryption

Related Theory:

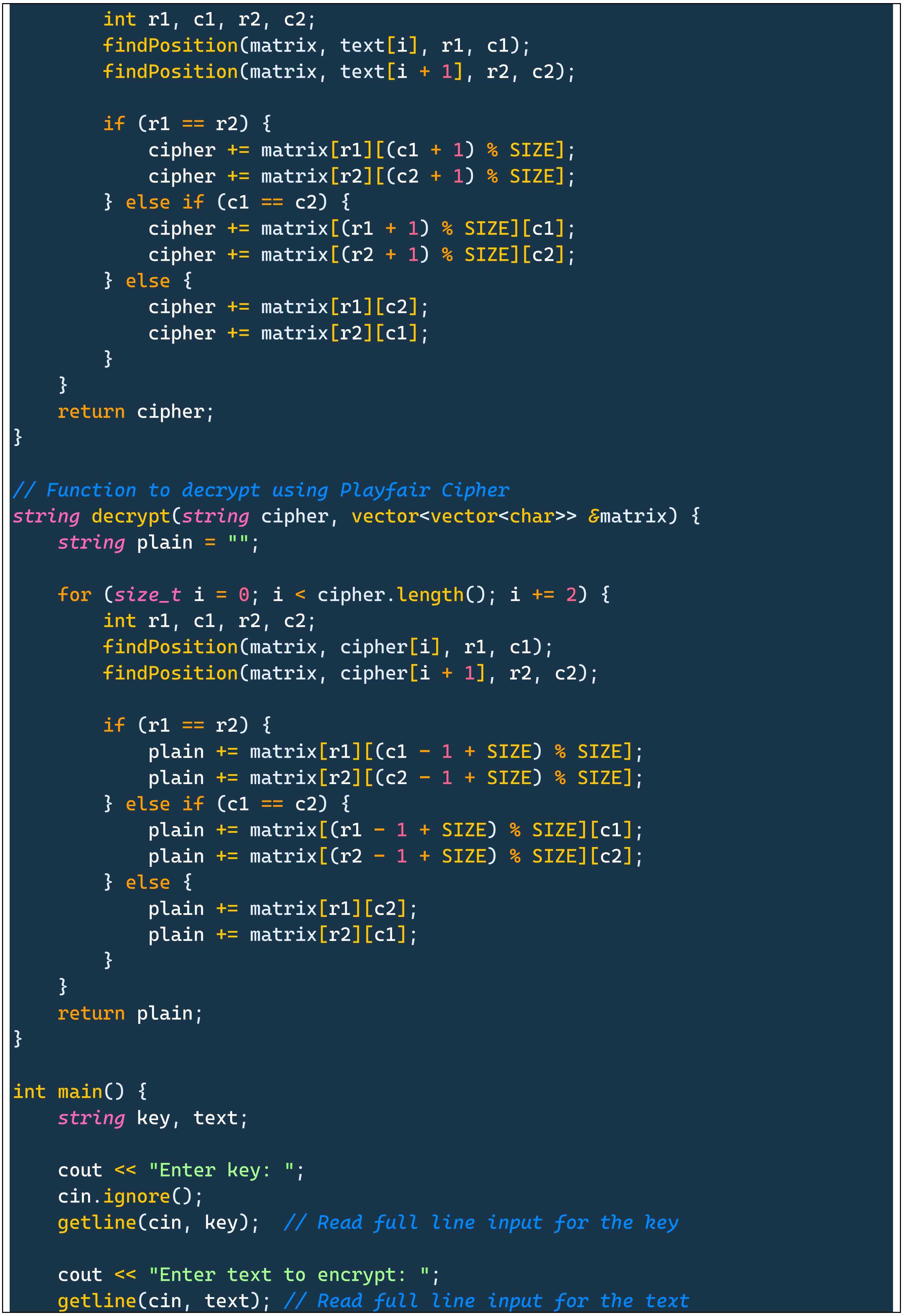
Implementation Details:

Playfair Cipher

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

#define SIZE 5

// Function to remove duplicate letters from the key
string removeDuplicates(string key) {
    string result = "";
    vector<bool> seen(26, false);
    for (char c : key) {
        if (c >= 'a' && c <= 'z') c -= 32; // Convert to uppercase
        if (c < 'A' || c > 'Z') continue;  // Ignore non-alphabetic
characters
        if (c == 'J') c = 'I'; // Treat 'J' as 'I'
        if (!seen[c - 'A']) {
            result += c;
            seen[c - 'A'] = true;
        }
    }
    return result;
}

// Function to generate the Playfair cipher matrix
vector<vector<char>> generateMatrix(string key) {
    key = removeDuplicates(key);
    vector<vector<char>> matrix(SIZE, vector<char>(SIZE, ' '));
    vector<bool> used(26, false);
    int row = 0, col = 0;

    for (char c : key) {
        if (!used[c - 'A']) {
            matrix[row][col++] = c;
            used[c - 'A'] = true;
            if (col == SIZE) {
                col = 0;
                row++;
            }
        }
    }

    for (char c = 'A'; c <= 'Z'; c++) {
        if (c == 'J') continue;
```

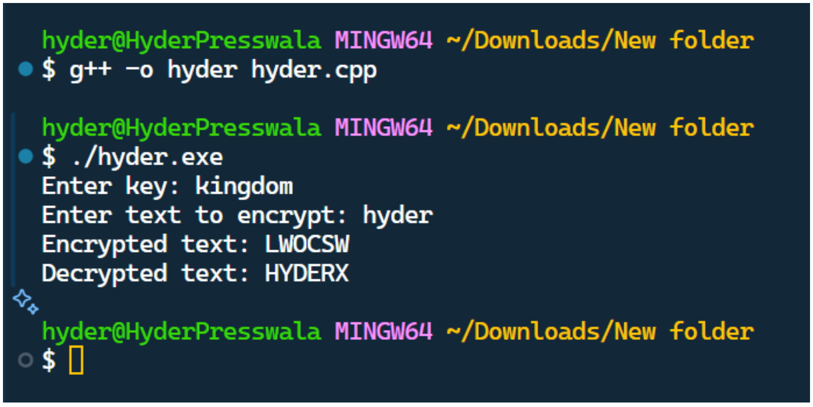


```
        if (!used[c - 'A']) {
            matrix[row][col++] = c;
            used[c - 'A'] = true;
            if (col == SIZE) {
                col = 0;
                row++;
            }
        }
    }
    return matrix;
}

// Function to find the position of a character in the matrix
void findPosition(vector<vector<char>> &matrix, char c, int &row, int &col) {
    if (c == 'J') c = 'I';
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (matrix[i][j] == c) {
                row = i;
                col = j;
                return;
            }
        }
    }
}

// Function to prepare the input text for encryption
string prepareText(string text) {
    string result = "";
    for (char c : text) {
        if (c >= 'a' && c <= 'z') c -= 32; // Convert to uppercase
        if (c < 'A' || c > 'Z') continue;  // Ignore non-alphabetic
characters
        if (c == 'J') c = 'I';
        result += c;
    }
    for (size_t i = 0; i < result.length() - 1; i += 2) {
        if (result[i] == result[i + 1])
            result.insert(i + 1, "X");
    }
    if (result.length() % 2 == 1) result += 'X';
    return result;
}

// Function to encrypt using Playfair Cipher
string encrypt(string text, vector<vector<char>> &matrix) {
    text = prepareText(text);
    string cipher = "";

    for (size_t i = 0; i < text.length(); i += 2) {
```



```
        int r1, c1, r2, c2;
        findPosition(matrix, text[i], r1, c1);
        findPosition(matrix, text[i + 1], r2, c2);

        if (r1 == r2) {
            cipher += matrix[r1][(c1 + 1) % SIZE];
            cipher += matrix[r2][(c2 + 1) % SIZE];
        } else if (c1 == c2) {
            cipher += matrix[(r1 + 1) % SIZE][c1];
            cipher += matrix[(r2 + 1) % SIZE][c2];
        } else {
            cipher += matrix[r1][c2];
            cipher += matrix[r2][c1];
        }
    }
    return cipher;
}

// Function to decrypt using Playfair Cipher
string decrypt(string cipher, vector<vector<char>> &matrix) {
    string plain = "";

    for (size_t i = 0; i < cipher.length(); i += 2) {
        int r1, c1, r2, c2;
        findPosition(matrix, cipher[i], r1, c1);
        findPosition(matrix, cipher[i + 1], r2, c2);

        if (r1 == r2) {
            plain += matrix[r1][(c1 - 1 + SIZE) % SIZE];
            plain += matrix[r2][(c2 - 1 + SIZE) % SIZE];
        } else if (c1 == c2) {
            plain += matrix[(r1 - 1 + SIZE) % SIZE][c1];
            plain += matrix[(r2 - 1 + SIZE) % SIZE][c2];
        } else {
            plain += matrix[r1][c2];
            plain += matrix[r2][c1];
        }
    }
    return plain;
}

int main() {
    string key, text;

    cout << "Enter key: ";
    cin.ignore();
    getline(cin, key);  // Read full line input for the key

    cout << "Enter text to encrypt: ";
    getline(cin, text); // Read full line input for the text
    vector<vector<char>> matrix = generateMatrix(key);

    string cipher = encrypt(text, matrix);
    cout << "Encrypted text: " << cipher << endl;

    string decrypted = decrypt(cipher, matrix);
    cout << "Decrypted text: " << decrypted << endl;

    return 0;
}
```

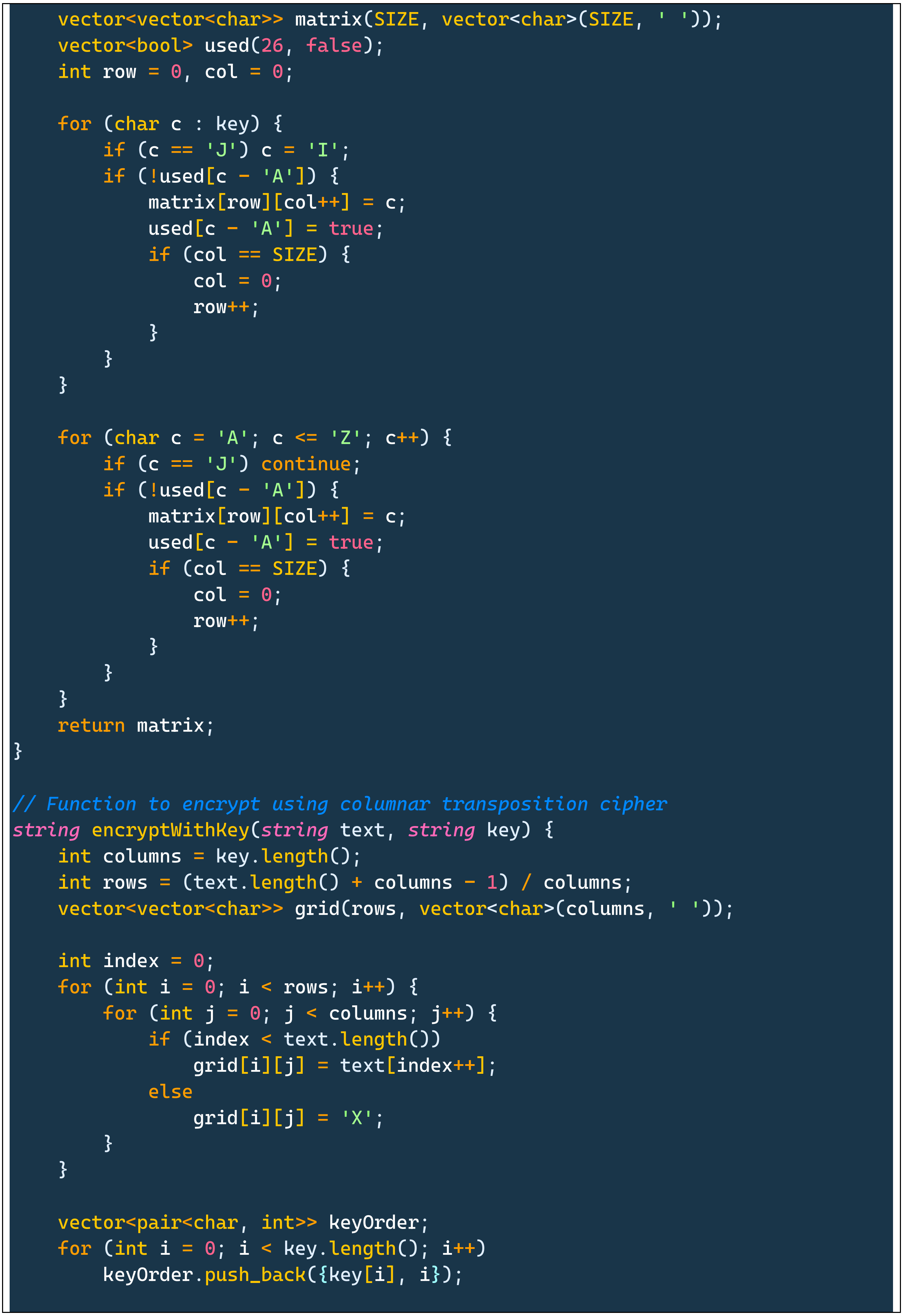
Output:-



```
hyder@HyderPresswala MINGW64 ~/Downloads/New folder
$ g++ -o hyder hyder.cpp

hyder@HyderPresswala MINGW64 ~/Downloads/New folder
$ ./hyder.exe
Enter key: kingdom
Enter text to encrypt: hyder
Encrypted text: LWOCSW
Decrypted text: HYDERX

hyder@HyderPresswala MINGW64 ~/Downloads/New folder
$ 
```

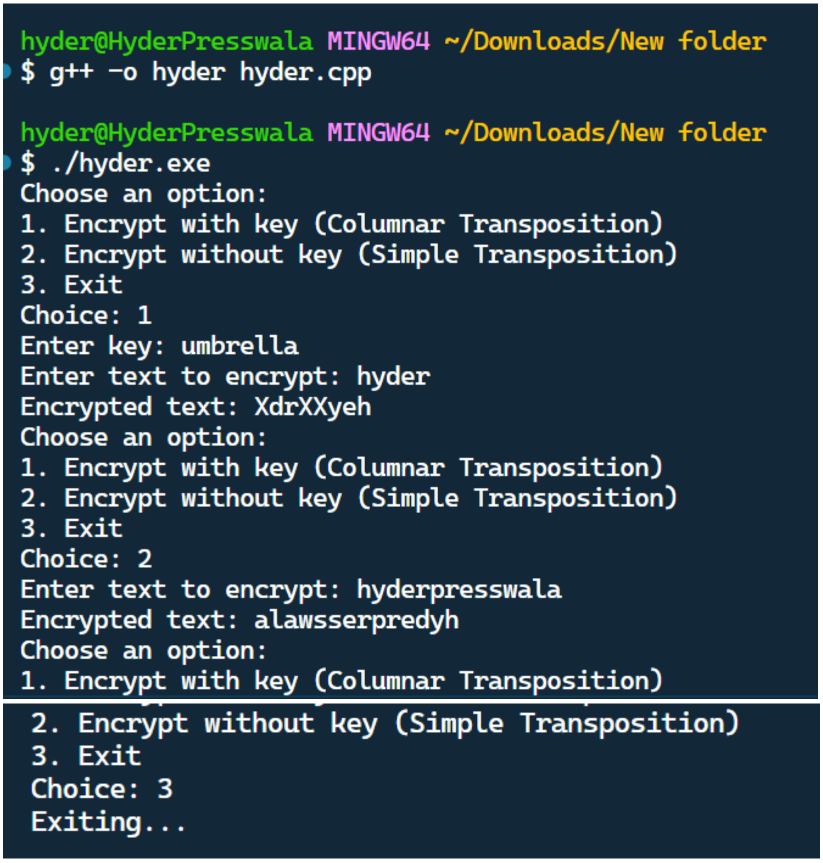
Transposition Cipher with key & without key

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

#define SIZE 5

// Function to remove duplicate letters from the key
string removeDuplicates(string key) {
    string result = "";
    vector<bool> seen(26, false);
    for (char c : key) {
        if (c == 'J') c = 'I'; // Treat 'J' as 'I'
        if (!seen[c - 'A']) {
            result += c;
            seen[c - 'A'] = true;
        }
    }
    return result;
}

// Function to generate the Playfair cipher matrix
vector<vector<char>> generateMatrix(string key) {
    key = removeDuplicates(key);
```



```
    vector<vector<char>> matrix(SIZE, vector<char>(SIZE, ' '));
    vector<bool> used(26, false);
    int row = 0, col = 0;

    for (char c : key) {
        if (c == 'J') c = 'I';
        if (!used[c - 'A']) {
            matrix[row][col++] = c;
            used[c - 'A'] = true;
            if (col == SIZE) {
                col = 0;
                row++;
            }
        }
    }

    for (char c = 'A'; c <= 'Z'; c++) {
        if (c == 'J') continue;
        if (!used[c - 'A']) {
            matrix[row][col++] = c;
            used[c - 'A'] = true;
            if (col == SIZE) {
                col = 0;
                row++;
            }
        }
    }
    return matrix;
}

// Function to encrypt using columnar transposition cipher
string encryptWithKey(string text, string key) {
    int columns = key.length();
    int rows = (text.length() + columns - 1) / columns;
    vector<vector<char>> grid(rows, vector<char>(columns, ' '));

    int index = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < columns; j++) {
            if (index < text.length())
                grid[i][j] = text[index++];
            else
                grid[i][j] = 'X';
        }
    }

    vector<pair<char, int>> keyOrder;
    for (int i = 0; i < key.length(); i++)
        keyOrder.push_back({key[i], i});
    sort(keyOrder.begin(), keyOrder.end());

    string cipher = "";
    for (auto &pair : keyOrder) {
        int col = pair.second;
        for (int i = 0; i < rows; i++)
            cipher += grid[i][col];
    }
    return cipher;
}

// Function to encrypt using simple transposition cipher (without key)
string encryptWithoutKey(string text) {
    reverse(text.begin(), text.end());
    return text;
}

int main() {
    int choice;
    string key, text;

    while (true) {
        cout << "Choose an option:\n1. Encrypt with key (Columnar Transposition)\n2. Encrypt without key (Simple Transposition)\n3. Exit\nChoice: ";
        cin >> choice;
        cin.ignore();

        if (choice == 1) {
            cout << "Enter key: ";
            getline(cin, key);
            cout << "Enter text to encrypt: ";
            getline(cin, text);
            cout << "Encrypted text: " << encryptWithKey(text, key) << endl;
        } else if (choice == 2) {
            cout << "Enter text to encrypt: ";
            getline(cin, text);
            cout << "Encrypted text: " << encryptWithoutKey(text) << endl;
        } else if (choice == 3) {
            cout << "Exiting..." << endl;
            break;
        } else {
            cout << "Invalid choice. Try again." << endl;
        }
    }
    return 0;
}
```

Output



```
hyder@HyderPresswala MINGW64 ~/Downloads/New folder
$ g++ -o hyder hyder.cpp

hyder@HyderPresswala MINGW64 ~/Downloads/New folder
$ ./hyder.exe
Choose an option:
1. Encrypt with key (Columnar Transposition)
2. Encrypt without key (Simple Transposition)
3. Exit
Choice: 1
Enter key: umbrella
Enter text to encrypt: hyder
Encrypted text: XdrXXyeh
Choose an option:
1. Encrypt with key (Columnar Transposition)
2. Encrypt without key (Simple Transposition)
3. Exit
Choice: 2
Enter text to encrypt: hyderpresswala
Encrypted text: alawsserpredyh
Choose an option:
1. Encrypt with key (Columnar Transposition)
2. Encrypt without key (Simple Transposition)
3. Exit
Choice: 3
Exiting...
```

Results/Output:

Conclusion:

Implemented transposition cipher for both encryption and decryption.

Post-Lab Questions:

1. **Write the points of difference between mono-alphabetic cipher and poly-alphabetic cipher.**

Mono-Alphabetic Cipher:

- Uses a single substitution rule for the entire message.
- One letter is always replaced by the same letter.
- Easier to crack using frequency analysis.

Poly-Alphabetic Cipher:

- Uses multiple substitution rules (multiple alphabets).
- A letter can be substituted by different letters depending on its position.
- More secure against frequency analysis.

2. **Explain the working of a rail-fence cipher with the help of an example.**

- The rail-fence cipher is a transposition cipher where the text is written in a zigzag pattern on a certain number of "rails."
- For example, with 3 rails and the message "HELLO", it would be written as:

H . . . O . . .

. E . L . L .

. . L . . .

The cipher text is then formed by reading the letters horizontally: "HOLELLO".

3. **Discuss any three applications of cryptography.**

- Data Security: Protects sensitive information in communication, e.g., online banking.
- Digital Signatures: Verifies the authenticity and integrity of documents.
- Secure Communication: Ensures privacy in email and instant messaging systems.