Practical no. 1(A)

Sahil Tiwaskar - 04B

<u>Caesar Cipher & Modified Caesar Cipher</u>

```
#include <bits/stdc++.h>
using namespace std;

string ceasarCipher(string text, int k, bool encrypt){
    string ans = "";
    for (auto ch : text){
        int val = int(ch) - 97;
        val = encrypt ? ((val + k) % 26) : ((val - k) % 26);
        ans +=  char(val + 65);
    }
    return ans;
}

int main(){
    string text = "";
    int k = 0;
    int ceasar = 0;
    int encryt = 0;

    cout << "Text: ";
    cin >> text;

    cout << "1 for Ceasar Cipher. "
         << "2 for Modified Ceasar Cipher: ";
    cin >> ceasar;

    if (ceasar == 1){
        k = 3;
    } else{
        cout << "Value of k: ";
        cin >> k;
    }

    cout << "1 for Encryption. "
         << "2 for Decryption: ";
    cin >> encryt;

    string ans = (encryt == 1)? ceasarCipher(text, k, true) : ceasarCipher(text, k, false);
    cout<<ans;

    cout << endl;
    return 0;
}
```

Caesar Cipher

```
Text: hello
1 for Ceasar Cipher. 2 for Modified Ceasar Cipher: 1
1 for Encryption. 2 for Decryption: 1
Final Text - KHOOR

Text: khoor
1 for Ceasar Cipher. 2 for Modified Ceasar Cipher: 1
1 for Encryption. 2 for Decryption: 2
Final Text - HELLO

Text: hello
1 for Ceasar Cipher. 2 for Modified Ceasar Cipher: 1
1 for Encryption. 2 for Decryption: 5
ERROR! Select 1 or 2 only
```

Modified Caesar Cipher

```
Text: hello
1 for Ceasar Cipher. 2 for Modified Ceasar Cipher: 2
Value of k: 15
1 for Encryption. 2 for Decryption: 1
Final Text - WTAAD

Text: wtaad
1 for Ceasar Cipher. 2 for Modified Ceasar Cipher: 2
Value of k: 15
1 for Encryption. 2 for Decryption: 2
Final Text - HELLO
```

Vigenere Cipher method

```
#include <bits/stdc++.h>
using namespace std;
void printTable(vector<char> pt, vector<int> pt_values, vector<int> k_values,
                vector<char> ct) {
    cout << "PT:      ";
    for (char c : pt) cout << c << " ";
    cout << endl;
    cout << "Value:   ";
    for (int val : pt_values) cout << setw(2) << val << " ";
    cout << endl;
    cout << "K Value: ";
    for (int val : k_values) cout << setw(2) << val << " ";
    cout << endl;
    cout << "CT:      ";
    for (char c : ct) cout << c << " ";
    cout << endl;
}
string vignereCipher(string text, string keyword, bool encrypt) {
    string ans = "";
    int n = text.size();
    int m = keyword.size();
    vector<char> pt(text.begin(), text.end());
    vector<int> pt_values;
    vector<int> k_values;
    vector<char> ct;
    for (int i = 0; i < n; i++) {
        int val = int(text[i] - 97);
        pt_values.push_back(val);
        int k = int(keyword[i % m] - 65);
        k_values.push_back(k);
        val = encrypt ? ((val + k) % 26) : ((val - k + 26) % 26);
        ct.push_back(char(val + 65));
    }
    printTable(pt, pt_values, k_values, ct);
    for (char c : ct) ans += c;
    return ans;
}
```

```
int main() {
    string text = "";
    string keyword = "";
    int encrypt = 0;
    cout << "Text: ";
    cin >> text;
    cout << "Keyword: ";
    cin >> keyword;
    cout << "1 for Encryption, 2 for Decryption: ";
    cin >> encrypt;

    transform(text.begin(), text.end(), text.begin(), ::tolower);
    transform(keyword.begin(), keyword.end(), keyword.begin(), ::toupper);
    string ans = (encrypt == 1) ? vignereCipher(text, keyword, true) : vignereCipher(text,
            keyword, false);
    cout << ans << endl;

    return 0;
}
```

Vigenere Cipher

```
Text: sheislistening
Keyword: PASCAL
1 for Encryption, 2 for Decryption: 1
PT:       s  h  e  i  s  l  i  s  t  e  n  i  n  g
Value:    18 07 04 08 18 11 08 18 19 04 13 08 13 06
K Value:  15 00 18 02 00 11 15 00 18 02 00 11 15 00
CT:       H  H  W  K  S  W  X  S  L  G  N  T  C  G
Final Text: HHWKSWXSLGNTCG


Text: sheislistening
Keyword: pascal
1 for Encryption, 2 for Decryption: 2
PT:       s  h  e  i  s  l  i  s  t  e  n  i  n  g
Value:    18 07 04 08 18 11 08 18 19 04 13 08 13 06
K Value:  15 00 18 02 00 11 15 00 18 02 00 11 15 00
CT:       D  H  M  G  S  A  T  S  B  C  N  X  Y  G
Final Text: DHMGSATSBCNXYG
```

Practical no. 1(B)

Sahil Tiwaskar - 04B

Playfair Cipher

```
#include <bits/stdc++.h>
using namespace std;

// Function to generate the key matrix
vector<vector<char>> generateKeyMatrix(const string& key) {
    string filteredKey;
    vector<vector<char>> matrix(5, vector<char>(5));
    vector<bool> used(26, false);

    // Process the key
    for (char c : key) {
        c = toupper(c);
        if (c == 'J') c = 'I';  // Treat 'J' as 'I'
        if (isalpha(c) && !used[c - 'A']) {
            filteredKey += c;
            used[c - 'A'] = true;
        }
    }

    // Fill filteredKey with remaining letters
    for (char c = 'A'; c <= 'Z'; ++c) {
        if (c == 'J') continue;
        if (!used[c - 'A']) {
            filteredKey += c;
        }
    }

    // Fill the matrix
    size_t index = 0;
    for (int i = 0; i < 5; ++i) {
        for (int j = 0; j < 5; ++j) {
            matrix[i][j] = filteredKey[index++];
        }
    }

    return matrix;
}

// Function to display the key matrix
void displayKeyMatrix(const vector<vector<char>>& matrix) {
    cout << "Key Matrix:" << endl;
    for (const auto& row : matrix) {
        for (char c : row) {
            cout << c << ' ';
```

```
        }
        cout << endl;
    }
}

// Function to format the text
string formatText(const string& text) {
    string formattedText;
    char prevChar = '\0';

    for (char c : text) {
        c = toupper(c);
        if (c == 'J') c = 'I';
        if (isalpha(c)) {
            if (c == prevChar) formattedText += 'X';
            formattedText += c;
            prevChar = c;
        }
    }

    if (formattedText.size() % 2 != 0) {
        formattedText += 'X';
    }

    return formattedText;
}

// Function to find the position of a character in the matrix
pair<int, int> findPosition(const vector<vector<char>>& matrix, char c) {
    for (int i = 0; i < 5; ++i) {
        for (int j = 0; j < 5; ++j) {
            if (matrix[i][j] == c) {
                return {i, j};
            }
        }
    }
    return {-1, -1};  // This should not happen if the matrix is correctly generated
}

// Function to encrypt or decrypt a pair of characters
string processPair(char a, char b, const vector<vector<char>>& matrix, bool encrypt) {
    auto [row1, col1] = findPosition(matrix, a);
    auto [row2, col2] = findPosition(matrix, b);

    if (row1 == row2) {
        // Same row
        return string(1, matrix[row1][(col1 + (encrypt ? 1 : 4)) % 5]) +
               matrix[row2][(col2 + (encrypt ? 1 : 4)) % 5];
    } else if (col1 == col2) {
```

```
        // Same column
        return string(1, matrix[(row1 + (encrypt ? 1 : 4)) % 5][col1]) +
               matrix[(row2 + (encrypt ? 1 : 4)) % 5][col2];
    } else {
        // Rectangle
        return string(1, matrix[row1][col2]) + matrix[row2][col1];
    }
}

// Function to perform Playfair cipher encryption or decryption
string playfairCipher(const string& text, const string& key, bool encrypt) {
    vector<vector<char>> matrix = generateKeyMatrix(key);
    string formattedText = formatText(text);
    string result;

    for (size_t i = 0; i < formattedText.size(); i += 2) {
        result += processPair(formattedText[i], formattedText[i + 1], matrix, encrypt);
    }

    return result;
}

int main() {
    string key, plaintext;

    // Get key and plaintext from the user
    cout << "Enter the key: ";
    getline(cin, key);
    cout << "Enter the plaintext: ";
    getline(cin, plaintext);

    // Remove spaces and punctuation from plaintext
    plaintext.erase(remove_if(plaintext.begin(), plaintext.end(), [](char c) {
        return !isalpha(c);
    }), plaintext.end());

    vector<vector<char>> matrix = generateKeyMatrix(key);
    displayKeyMatrix(matrix);

    string ciphertext = playfairCipher(plaintext, key, true);
    string decryptedText = playfairCipher(ciphertext, key, false);

    cout << "Plaintext: " << plaintext << endl;
    cout << "Ciphertext: " << ciphertext << endl;
    cout << "Decrypted: " << decryptedText << endl;

    return 0;
}
```

PlayFair Cipher

```
Enter the key: MONARCHY
Enter the plaintext: instrumentsz
Key Matrix:
M O N A R
C H Y B D
E F G I K
L P Q S T
U V W X Z
Plaintext: instrumentsz
Ciphertext: GATLMZCLRQTX
Decrypted: INSTRUMENTSZ
```