Information Security

Name: KANISH
College Roll No: CSC/21/53
University Roll No: 21059570017

4. Implement monoalphabetic and polyalphabetic cipher substitution operation.

```
#include <iostream>
#include <string>
#include <cctype> // for isalpha
using namespace std;

// Function to perform monoalphabetic encryption (Caesar cipher)
string monoalphabeticEncrypt(string plaintext, int key) {
    string ciphertext = "";

    for (size_t i = 0; i < plaintext.length(); ++i) {
        char c = plaintext[i];
        if (isalpha(c)) {
            char base = islower(c) ? 'a' : 'A';
            char shifted = (c - base + key) % 26 + base;
            ciphertext += shifted;
        } else {
            ciphertext += c; // Leave non-alphabetic characters
unchanged
        }
    }

    return ciphertext;
}

// Function to perform monoalphabetic decryption (Caesar cipher)
string monoalphabeticDecrypt(string ciphertext, int key) {
    return monoalphabeticEncrypt(ciphertext, 26 - key); // Decryption
is just encryption with inverse key
}
```

```
// Function to perform polyalphabetic encryption (Vigenère cipher)
string polyalphabeticEncrypt(string plaintext, string keyword) {
    string ciphertext = "";
    int keywordLen = keyword.length();
    int index = 0;

    for (size_t i = 0; i < plaintext.length(); ++i) {
        char c = plaintext[i];
        if (isalpha(c)) {
            char base = islower(c) ? 'a' : 'A';
            char shifted = (c - base + (keyword[index % keywordLen] -
'a')) % 26 + base;
            ciphertext += shifted;
            index++;
        } else {
            ciphertext += c; // Leave non-alphabetic characters
unchanged
        }
    }

    return ciphertext;
}

// Function to perform polyalphabetic decryption (Vigenère cipher)
string polyalphabeticDecrypt(string ciphertext, string keyword) {
    string plaintext = "";
    int keywordLen = keyword.length();
    int index = 0;

    for (size_t i = 0; i < ciphertext.length(); ++i) {
        char c = ciphertext[i];
        if (isalpha(c)) {
            char base = islower(c) ? 'a' : 'A';
            char shifted = (c - base - (keyword[index % keywordLen] -
'a') + 26) % 26 + base;
            plaintext += shifted;
            index++;
        } else {
            plaintext += c; // Leave non-alphabetic characters
unchanged
        }
    }

    return plaintext;
}
```

```
int main() {
    string plaintext;
    int key;
    string keyword;

    cout << "Enter plaintext: ";
    getline(cin, plaintext);

    cout << "Enter key for monoalphabetic encryption (Caesar cipher):
";
    cin >> key;
    cin.ignore(); // Clear input buffer

    cout << "Enter keyword for polyalphabetic encryption (Vigenère
cipher): ";
    getline(cin, keyword);

    // Monoalphabetic cipher (Caesar cipher)
    string encryptedMonoalphabetic = monoalphabeticEncrypt(plaintext,
key);
    string decryptedMonoalphabetic =
monoalphabeticDecrypt(encryptedMonoalphabetic, key);

    cout << "\nMonoalphabetic (Caesar cipher) Encryption:" << endl;
    cout << "Ciphertext: " << encryptedMonoalphabetic << endl;
    cout << "Decrypted text: " << decryptedMonoalphabetic << endl <<
endl;

    // Polyalphabetic cipher (Vigenère cipher)
    string encryptedPolyalphabetic = polyalphabeticEncrypt(plaintext,
keyword);
    string decryptedPolyalphabetic =
polyalphabeticDecrypt(encryptedPolyalphabetic, keyword);

    cout << "Polyalphabetic (Vigenère cipher) Encryption:" << endl;
    cout << "Ciphertext: " << encryptedPolyalphabetic << endl;
    cout << "Decrypted text: " << decryptedPolyalphabetic << endl;

    return 0;
}
```

```
Enter plaintext: KANISH
Enter key for monoalphabetic encryption (Caesar cipher): 3
Enter keyword for polyalphabetic encryption (VigenΦre cipher): KEY

Monoalphabetic (Caesar cipher) Encryption:
Ciphertext: NDQLVK
Decrypted text: KANISH

Polyalphabetic (VigenΦre cipher) Encryption:
Ciphertext: 5?F37@
Decrypted text: 5?B37@

--------------------------------
Process exited after 12.02 seconds with return value 0
Press any key to continue . . .
```