NS 3 Caesar cipher

```
#include <iostream>
#include <string>
#include <algorithm>

// Function to perform Caesar cipher encryption
std::string caesarCipherEncrypt(std::string text) {
    for (char& c : text) {
        if (isalpha(c)) {
            if (islower(c)) {
                c = 'a' + (c - 'a' + 3) % 26;
            } else {
                c = 'A' + (c - 'A' + 3) % 26;
            }
        }
    }
    return text;
}

// Function to perform Caesar cipher decryption
std::string caesarCipherDecrypt(std::string text) {
    for (char& c : text) {
        if (isalpha(c)) {
            if (islower(c)) {
                c = 'a' + (c - 'a' - 3 + 26) % 26;
            } else {
                c = 'A' + (c - 'A' - 3 + 26) % 26;
            }
        }
    }
    return text;
```

```
}

int main() {
    std::string text;
    int choice;

    std::cout << "Please enter a string: ";
    std::getline(std::cin, text);

    std::cout << "Please choose an option:\n";
    std::cout << "1. Encrypt the string\n";
    std::cout << "2. Decrypt the string\n";
    std::cin >> choice;

    std::string result;
    switch (choice) {
    case 1:
        result = caesarCipherEncrypt(text);
        std::cout << "Encrypted string: " << result << "\n";
        break;
    case 2:
        result = caesarCipherDecrypt(text);
        std::cout << "Decrypted string: " << result << "\n";
        break;
    default:
        std::cout << "Invalid choice\n";
        break;
    }

    return 0;
}
```

substitution cipher

```
#include <iostream>
#include <string> // Include the <string> header

// Function to perform substitution cipher encryption
std::string substitutionCipherEncrypt(std::string text) {
    std::string alphabet = "abcdefghijklmnopqrstuvwxyz";
    std::string key = "xyzabcdefghijklmnopqrstuvw";
    for (char& c : text) {
        if (isalpha(c)) {
            if (islower(c)) {
                c = key[c - 'a'];
            } else {
                c = toupper(key[tolower(c) - 'a']);
            }
        }
    }
    return text;
}

// Function to perform substitution cipher decryption
std::string substitutionCipherDecrypt(std::string text) {
    std::string alphabet = "abcdefghijklmnopqrstuvwxyz";
    std::string key = "xyzabcdefghijklmnopqrstuvw";
    for (char& c : text) {
        if (isalpha(c)) {
            if (islower(c)) {
                c = alphabet[key.find(c)];
            } else {
                c = toupper(alphabet[key.find(tolower(c))]);
            }
```

```
        }
    }
    return text;
}

int main() {
    std::string text;
    int choice;

    std::cout << "Please enter a string: ";
    std::getline(std::cin, text);

    std::cout << "Please choose an option:\n";
    std::cout << "1. Encrypt the string\n";
    std::cout << "2. Decrypt the string\n";
    std::cin >> choice;

    std::string result;
    switch (choice) {
        case 1:
            result = substitutionCipherEncrypt(text);
            std::cout << "Encrypted string: " << result << "\n";
            break;
        case 2:
            result = substitutionCipherDecrypt(text);
            std::cout << "Decrypted string: " << result << "\n";
            break;
        default:
            std::cout << "Invalid choice\n";
            break;
    }
```

```
    return 0;}
```

Transposition

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>

// Function to perform transposition cipher encryption
std::string transpositionCipherEncrypt(std::string text) {
    int key = 3; // Example key, can be any positive integer
    int len = text.length();
    int numRows = ceil((double)len / key);
    char matrix[numRows][key];

    int index = 0;
    for (int i = 0; i < numRows; i++) {
        for (int j = 0; j < key; j++) {
            if (index < len)
                matrix[i][j] = text[index++];
            else
                matrix[i][j] = ' '; // Padding with spaces if needed
        }
    }

    std::string encryptedText = "";
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < numRows; j++) {
            encryptedText += matrix[j][i];
        }
    }
```

```
    return encryptedText;
}

// Function to perform transposition cipher decryption
std::string transpositionCipherDecrypt(std::string text) {
    int key = 3; // Example key, should match the encryption key
    int len = text.length();
    int numRows = ceil((double)len / key);
    char matrix[numRows][key];

    int index = 0;
    for (int i = 0; i < key; i++) {
        for (int j = 0; j < numRows; j++) {
            matrix[j][i] = text[index++];
        }
    }

    std::string decryptedText = "";
    for (int i = 0; i < numRows; i++) {
        for (int j = 0; j < key; j++) {
            decryptedText += matrix[i][j];
        }
    }

    return decryptedText;
}

int main() {
    std::string input;
    std::cout << "Please enter a string: ";
```

```
    std::getline(std::cin, input);

    std::string transpositionEncrypted = transpositionCipherEncrypt(input);
    std::string transpositionDecrypted = transpositionCipherDecrypt(transpositionEncrypted);

    std::cout << "Transposition Cipher Encrypted: " << transpositionEncrypted << std::endl;
    std::cout << "Transposition Cipher Decrypted: " << transpositionDecrypted << std::endl;

    return 0;
}
```