Assignment 3

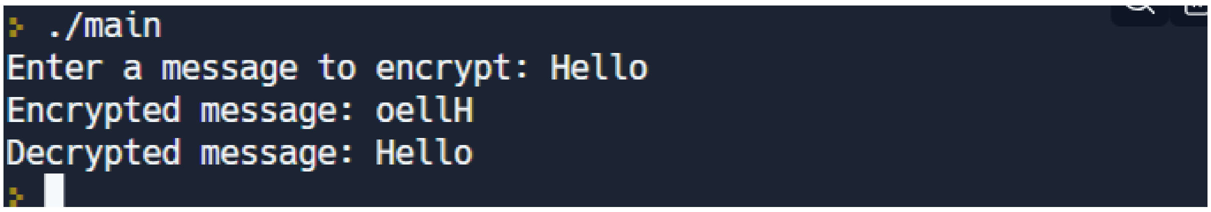
Personalized Encryption/Decryption Algorithm:

Shuffle Encryption/Decryption Algorithm

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

string shuffleEncrypt(const string &message) {
   string encryptedMessage = message;
   vector<char> shufflePattern(message.begin(), message.end());

   // Shuffle the characters randomly
   srand(time(nullptr));
   random_shuffle(shufflePattern.begin(), shufflePattern.end());

   for (size_t i = 0; i < message.length(); ++i) {
      encryptedMessage[i] = shufflePattern[i];
   }

   return encryptedMessage;
}

string shuffleDecrypt(const string &encryptedMessage) {
   string decryptedMessage = encryptedMessage;
   vector<char> shufflePattern(encryptedMessage.begin(), encryptedMessage.end());
```

```
    // Sort the characters to reverse the shuffle
    sort(shufflePattern.begin(), shufflePattern.end());

    for (size_t i = 0; i < encryptedMessage.length(); ++i) {
        decryptedMessage[i] = shufflePattern[i];
    }

    return decryptedMessage;
}

int main() {
    string message;
    cout << "Enter a message to encrypt: ";
    getline(cin, message);

    // Encrypt the message using shuffle
    string encryptedMessage = shuffleEncrypt(message);
    cout << "Encrypted message: " << encryptedMessage << endl;

    // Decrypt the message using shuffle
    string decryptedMessage = shuffleDecrypt(encryptedMessage);
    cout << "Decrypted message: " << decryptedMessage << endl;

    return 0;
}
```

Output:



```
> ./main
Enter a message to encrypt: Hello
Encrypted message: oellH
Decrypted message: Hello
> 
```

Conclusion:

The Shuffle Encryption/Decryption Algorithm is a basic technique that introduces variability to a message by rearranging its characters. During encryption, a random shuffle pattern is generated, and characters in the message are shuffled accordingly. This pattern is crucial for decryption, where the reverse of the shuffle pattern is applied to reconstruct the original message. While this algorithm offers simplicity and customization through randomness, it lacks the security measures necessary for protecting sensitive data. Therefore, it is essential to choose encryption methods that have undergone rigorous scrutiny and testing in real-world applications to ensure data confidentiality and integrity.