Name - Khushi Nitinkumar Patel
PRN - 2020BTECS00037
Batch - B3

Assignment no 4: Implementation of Vigenere Cipher

Introduction

Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenère square or Vigenère table.

Encryption:

Suppose we want to encrypt the plaintext message "HELLO" using the keyword "KEY."

1. Key Expansion:

- Repeat the keyword to match the length of the plaintext message:

Plaintext: H E L L O
Keyword: K E Y K E

2. Letter to Number Conversion:

- Convert the letters of the plaintext and the keyword to their corresponding numerical values (A=0, B=1, ..., Z=25):

Plaintext: 7 4 11 11 14
Keyword: 10 4 24 10 4

3. Encryption:

- Add the corresponding values of the plaintext and the keyword, taking care to wrap around the alphabet if the sum is greater than 25:

Ciphertext: 17 8 9 21 18

4. Number to Letter Conversion:

- Convert the numerical values back to letters:

Ciphertext: R I J V S

So, "HELLO" encrypted with the keyword "KEY" becomes "RIJVS."

Decryption

Now, let's decrypt the ciphertext "RIJVS" using the same keyword "KEY."

1. Key Expansion:

- Repeat the keyword to match the length of the ciphertext:

Ciphertext: R I J V S
Keyword: K E Y K E

2. Letter to Number Conversion:

- Convert the letters of the ciphertext and the keyword to their corresponding numerical values (A=0, B=1, ..., Z=25):

Ciphertext: 17 8 9 21 18
Keyword: 10 4 24 10 4

3. Decryption:

- Subtract the corresponding values of the keyword from the ciphertext, taking care to wrap around the alphabet if the difference is negative:

Plaintext: 7 4 11 11 14

4. Number to Letter Conversion:

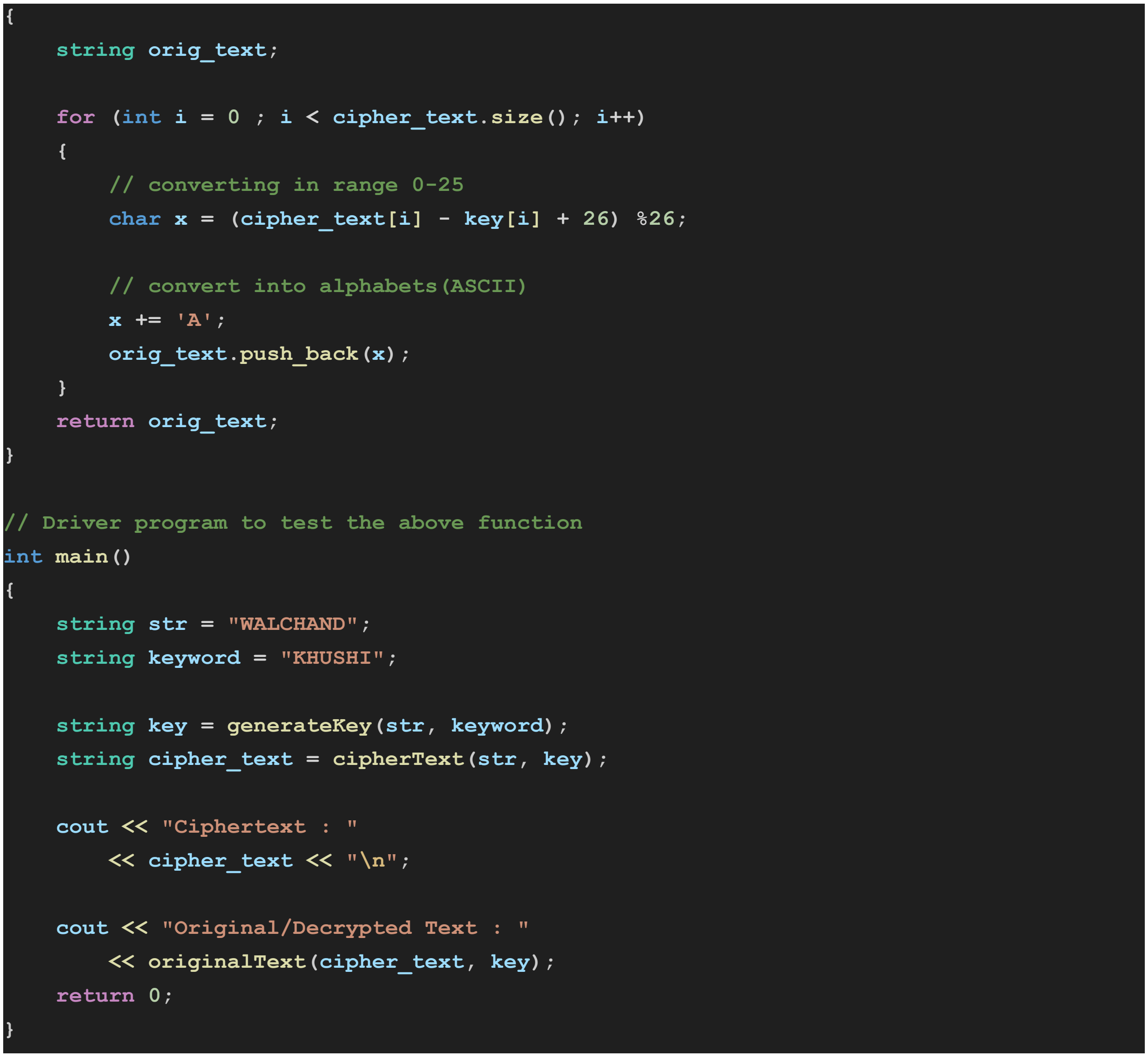
- Convert the numerical values back to letters:

Plaintext: H E L L O

So, "RIJVS" decrypted with the keyword "KEY" becomes "HELLO."

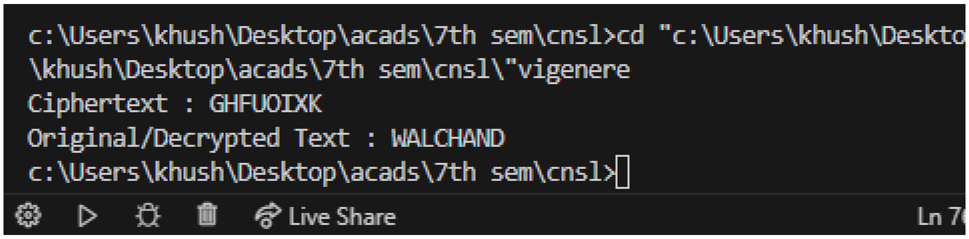
Encryption and Decryption Code:

```cpp
// C++ code to implement Vigenere Cipher
#include<bits/stdc++.h>
using namespace std;

// This function generates the key in
// a cyclic manner until it's length isn't
// equal to the length of original text
string generateKey(string str, string key)
{
    int x = str.size();

    for (int i = 0; ; i++)
    {
        if (x == i)
            i = 0;
        if (key.size() == str.size())
            break;
        key.push_back(key[i]);
    }
    return key;
}

// This function returns the encrypted text
// generated with the help of the key
string cipherText(string str, string key)
{
    string cipher_text;

    for (int i = 0; i < str.size(); i++)
    {
        // converting in range 0-25
        char x = (str[i] + key[i]) %26;

        // convert into alphabets(ASCII)
        x += 'A';

        cipher_text.push_back(x);
    }
    return cipher_text;
}

// This function decrypts the encrypted text
// and returns the original text
string originalText(string cipher_text, string key)
```



```
{
    string orig_text;

    for (int i = 0 ; i < cipher_text.size(); i++)
    {
        // converting in range 0-25
        char x = (cipher_text[i] - key[i] + 26) %26;

        // convert into alphabets(ASCII)
        x += 'A';
        orig_text.push_back(x);
    }
    return orig_text;
}

// Driver program to test the above function
int main()
{
    string str = "WALCHAND";
    string keyword = "KHUSHI";

    string key = generateKey(str, keyword);
    string cipher_text = cipherText(str, key);

    cout << "Ciphertext : "
        << cipher_text << "\n";

    cout << "Original/Decrypted Text : "
        << originalText(cipher_text, key);
    return 0;
}
```

Output:



```
c:\Users\khush\Desktop\acads\7th sem\cnsl>cd "c:\Users\khush\Deskto
\khush\Desktop\acads\7th sem\cnsl\"vigenere
Ciphertext : GHFUOIXK
Original/Decrypted Text : WALCHAND
c:\Users\khush\Desktop\acads\7th sem\cnsl>
```