Vigenere Cipher

- formulated by Leon Battista Alberti around 1467
- Based on Vigenère square (Vigenère table) and Keyword.
- The keyword written along the plaintext.
- No space between words.
- The ciphertext will be the result of the cross point between each letter in plaintext and keyword letter according to Vigenère table.

- Example:

Text = ATTACK AT DAWN

Key=LEMON

Text	A	T	T	A	C	K	A	T	D	A	W	N
Key	L	E	M	O	N	L	E	M	O	N	L	E
Cipher	L	X	F	O	P	V	E	F	R	N	H	R

- The column for the plaintext and the row for keyword.
- For decryption do the reverse operation.
- **Example:** decrypt the following ciphertext with keyword "LEMON".

Ciphertext = LXFOPVEFRNHR

The result appear by finding the letter in keyword at the row then finding the letter in ciphertext at the same row, then, text letter is the letter at the top of column.

Cipher	L	X	F	O	P	V	E	F	R	N	H	R
Key	L	E	M	O	N	L	E	M	O	N	L	E
Text	A	T	T	A	C	K	A	T	D	A	W	N

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

- Vigenère can also be viewed algebraically.
- If the letters A–Z are taken to be the numbers 0–25, and addition is performed modulo 26, then Vigenère encryption using the key can be written:

$$C_i = E_K(M_i) = (M_i + K_i) \mod 26$$

- To encrypt Letter 'C' with key letter 'L' the calculation would result in (2+11) mod 26 =13 which equal letter 'n'.

- **Decryption:** for decryption use the following:

$$M_i = D_K(C_i) = (C_i - K_i) \mod 26$$

Example: you have cipher letter 'R' with key letter 'H' find the text Letter. (17-7) mod 26 = 10 which equal letter 'K' .

```
// C++ code to implement Vigenere Cipher
#include<iostream>
#include<string>
using namespace std;

// This function generates the key in
// a cyclic manner until it's length is
// 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 = "HELLOCIHAN";
	string keyword = "LOVE";

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

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

	cout << "OriginalText : "
		<< originalText(cipher_text, key)<<"\n";
	system("pause");
	return 0;
}
```

OUTPUT

???

note

*push_front() function is used to push elements into a list from the front. The new value is inserted into the list at the beginning, before the current first element and the container size is increased by 1.

Syntax :
listname.push_front(value)
Parameters :
The value to be added in the front is
passed as the parameter
Result :
Adds the *value* mentioned as the parameter
to the front of the list named as *listname*

example:
Input : list list{1, 2, 3, 4, 5};

list.push_front(6);

Output : 6, 1, 2, 3, 4, 5

Input : list list{5, 4, 3, 2, 1};

list.push_front(6);

Output :6, 5, 4, 3, 2, 1

```
CPP program to illustrate
// push_front() function
#include <iostream>
#include <list>
using namespace std;

int main()
{
    list<int> mylist{ 1, 2, 3, 4, 5 };
    mylist.push_front(6);

    // list becomes 6, 1, 2, 3, 4, 5

    for (auto it = mylist.begin(); it != mylist.end(); ++it)
```

```
        cout << ' ' << *it;
}
```

```
6 1 2 3 4 5
```

* push_back() function is used to push elements into a list from the back. The new value is inserted into the list at the end, after the current last element and the container size is increased by 1.

Syntax :
listname.push_back(value)
Parameters :
The value to be added in the back is
passed as the parameter
Result :
Adds the *value* mentioned as the parameter
to the back of the list named as *listname*

Examples:

```
Input : list list{1, 2, 3, 4, 5};
        list.push_back(6);
Output :1, 2, 3, 4, 5, 6

Input : list list{5, 4, 3, 2, 1};
        list.push_front(0);
Output :5, 4, 3, 2, 1, 0
```

```
// CPP program to illustrate

// push_back() function

#include <iostream>
```

```
#include <list>

using namespace std;



int main()

{

    list<int> mylist{ 1, 2, 3, 4, 5 };

    mylist.push_back(6);



    // list becomes 1, 2, 3, 4, 5, 6



    for (auto it = mylist.begin(); it != mylist.end(); ++it)

        cout << ' ' << *it;

}
```

Output:

```
1 2 3 4 5 6
```