Substitution Ciphers/ Caesar Cipher

- Use a correspondence table with which to substitute a character or symbol for each character of the original message.
- It is a simple and acceptable way of encrypting text.

The Caesar Cipher technique is one of the earliest and simplest methods of encryption technique. It's simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.
Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.
The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme,

A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

A shift may be any amount, so that general Caesar algorithm is

$$C = E(K, P) = (P + K) \bmod 26$$

Where K takes on a value in the range 1 to 25 . The decryption algorithm is simply

$$P = D(K, C) = (C - K) \bmod 26$$

The Caesar involves replacing each letter of the alphabet with the letter standing three place further down the alphabet.

$$C = E\ (3,P) = (P+3) \bmod 26$$

Plaintext Alphabet	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
Ciphertext Alphabet	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

Let us assign a numerical equivalent to each letter.

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Plain text: meet me after the toga party

Cipher text: PHHW PH DIWHU WKH WRJD SDUWB

```
// A C++ program to illustrate Caesar Cipher Technique
#include<iostream>
#include<string>
using namespace std;

 // This function receives text and shift and
// returns the encrypted text
string encrypt(string text, int s)
{
    string result = "";
     // traverse text
    for (int i=0;i<text.length();i++)
    {
        // apply transformation to each character
        // Encrypt Uppercase letters
        if (isupper(text[i]))
            result += char(int(text[i]+s-65)%26 +65);
     // Encrypt Lowercase letters
    else
        result += char(int(text[i]+s-97)%26 +97);
```

```
    }

   // Return the resulting string

    return result;

}

// Driver program to test the above function

int main()

{

    string text="ATTACKATONCE";

    int s = 4;

    cout << "Text : " << text;

    cout << "\nShift: " << s;

    cout << "\nCipher: " << encrypt(text, s);

system("pause");

    return 0;

}
```

Output:

```
Text : ATTACKATONCE

Shift: 4

Cipher: EXXEGOEXSRGI
```

How to decrypt?

//A C++ program to illustrate Caesar Cipher Technique

```
#include<iostream>
#include<string>
using namespace std;
//This function receives text and shift and returns the encrypted text
string encrypt(string text,int s)
{
	string result="";
	//traverse text
	for (int i=0;i<text.length();i++)
	{
		//apply transformation to each character
		//Encrypt Uppercase letters
		if(isupper(text[i]))
			result+=char(int(text[i]+s-65)%26 +65);
	//Encrypt Lowercase letters
	else
result+=char(int(text[i]+s-97)%26 +97);
	}
	//Return the resulting string
	return result;
}
```

```
//Driver program to test the above function

int main()

{

        string text="EXXEGOEXSRGI";

        int s = 4;

        cout<<"Text :"<<text;

        cout<<"\nShift:" << s;

        s = s%26; // ensuring that s lies between 0-25

        cout<<"\nCipher:"<<encrypt(text, 26-s);

system(“pause”);

        return 0;

}
```

Output:Copy

```
Text :EXXEGOEXSRGI
Shift:4
Cipher:ATTACKATONCE
```

How the `char(int(text[i]+s-65)%26 +65);` works

Key Fact - The ASCII code of the letter"A" is 65.

Here is how your cypher works - the *original expression* in the question title.

1. Take the ASCII value of a letter, subtract the value of "A" from it giving you a 0 based number.
2. Add the key value to this number shifting it by k places.
3. Now divide the number you got above by 26, **discard** the quotient and use the remainder. This is the modulo operator %. This always keeps you numbers in the 0-25 range, since dividing by 26 will never a have a remainder great than 25.
4. Add 65 to it to convert it into an "encrypted" uppercase letter.

This allows the key to be ANY number and still keeps the "encrypted" output within the ASCII range of A-Z.

Don't interpret the % operator as division. In reality, it's modulo or forget-the-quotient-I want-the-remainder operator.

Example

1. 0%2 is 0
2. 1%2 is 1
3. 2%2 is 0
4. 3%2 is 1