**Title: Study and implement a program for Caeser Cipher**

**Original Approach**

**Introduction:**

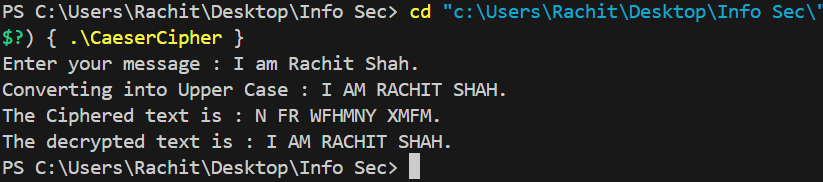
Caeser Cipher is one of the most simple and easiest Cryptography encryption technique. It is a type of Substitution Cipher. In this technique the message/plaintext is shifted by a certain number of places up or down in the alphabet system. The shift value by which the text is to be shifted is denoted as a key.

**Example:** I AM RACHIT SHAH.  
Let, key =5.  
Encrypted message: N FR WFHMNY XMFM.

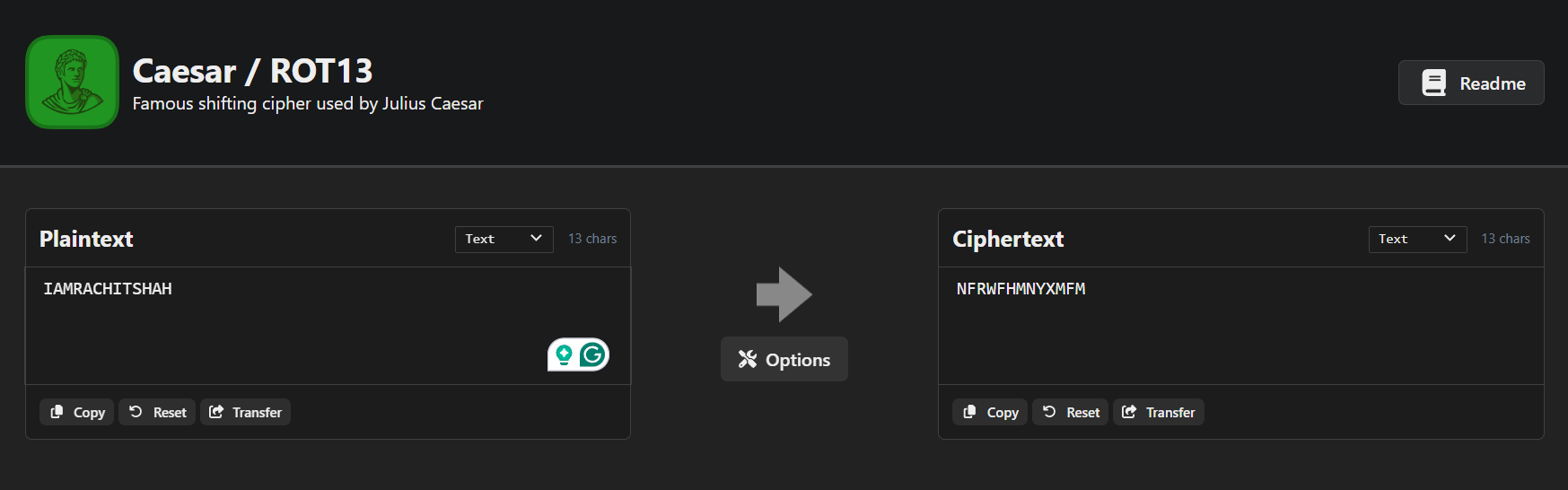
**Source Code:**

|  |
| --- |
| **#include <stdio.h>**  **#include <string.h>**  **#include <stdlib.h>**  //Globally declared variables  **char encrypted[100];**  **char decrypted[100];**  **int key = 5;**  //Function to encrypt the message  **void encrypt(char message[], char alphabets[], char encrypted[]) {**  **int length = strlen(message);**  **for (int i = 0; i < length; i++) {**  **char c = message[i];**  **int found = 0;**  **for (int j = 0; j < 26; j++) {**  **if (c == alphabets[j]) {**  **int temp = (j + key + 26) % 26;**  **encrypted[i] = alphabets[temp];**  **found = 1;**  **break;**  **}**  **}**      //Condition for SPACE  **if (!found) {**  **encrypted[i] = c;**  **}**  **}**  //Printing of encrypted text  **encrypted[length] = '\0';**  **printf("%s\n", encrypted);**  **}**  //Function to decrypt the encrypted text  **void decrypt(char encrypted[], char alphabets[], char decrypted[]){**  **int length = strlen(encrypted);**  **for(int i=0; i<length; i++){**  **char c = encrypted[i];**  **int found = 0;**  **for(int j=0; j<26; j++){**  **if (c == alphabets[j]) {**  **int temp = (j - key + 26) % 26;** //edge case condition also **tackled**  **decrypted[i] = alphabets[temp];**  **found = 1;**  **break;**  **}**  **}**          //Condition to handle SPACE  **if (!found) {**  **decrypted[i] = c;**  **}**  **}**      //Printing the decrypted text  **decrypted[length = '\0'];**  **printf("%s\n", decrypted);**  **}**  **void main(){**  **char alpha[26] = {'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'};**  **char message[100];**  **printf("Enter your message : ");**  **gets(message);**      //Converting the text into upper case for uniformity  **strupr(message);**  **printf("Converting into Upper Case : \0");**  **printf(message);**  **printf("\nThe Ciphered text is : \0");**  **encrypt(message,alpha,encrypted);**  **printf("The decrypted text is : \0");**  **decrypt(encrypted,alpha,decrypted);**  **}** |

**Output Screenshot:**

****

**Crypt-tool Output:**

****

**Revised Approach**

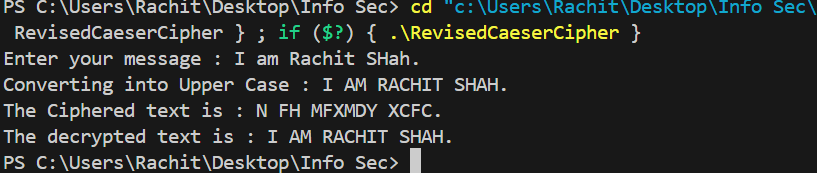
**Introduction :** Usually, all the characters are shifted as per the key value either to the right(up) or to the left(down). In this revised approach all the characters at the even position as moved right and the one at odd position are moved left.

**Example :** I AM RACHIT SHAH.  
Let, key =5.  
Encrypted message : N FH MFXMDY XCFC.

**Source Code:**

|  |
| --- |
| **#include <stdio.h>**  **#include <string.h>**  **#include <stdlib.h>**  //Globally declared variables  **char encrypted[100];**  **char decrypted[100];**  **int key = 5;**  //Function to encrypt the message  **void encrypt(char message[], char alphabets[], char encrypted[]) {**  **int length = strlen(message);**  **for (int i = 0; i < length; i++) {**  **char c = message[i];**  **int found = 0;**  **for (int j = 0; j < 26; j++) {**            //Condition to encrypt even characters  **if (c == alphabets[j] && i%2==0) {**  **int temp = (j + key) % 26;**  **encrypted[i] = alphabets[temp];**  **found = 1;**  **break;**  **}**            //Condition to encrypt even characters  **else if(c == alphabets[j] && i%2!=0){**  **int temp = (j - key + 26) % 26;**  **encrypted[i] = alphabets[temp];**  **found = 1;**  **break;**  **}**  **}**          //Condition for SPACE  **if (!found) {**  **encrypted[i] = c;**  **}**  **}**      //Printing of encrypted text  **encrypted[length] = '\0';**  **printf("%s\n", encrypted);**  **}**  //Function to decrypt the encrypted text  **void decrypt(char encrypted[], char alphabets[], char decrypted[]){**  **int length = strlen(encrypted);**  **for(int i=0; i<length; i++){**  **char c = encrypted[i];**  **int found = 0;**  **for(int j=0; j<26; j++){**      //Condition to decrypt even characters  **if (c == alphabets[j] && i%2==0) {**  **int temp = (j - key + 26) % 26;**  **decrypted[i] = alphabets[temp];**  **found = 1;**  **break;**  **}**      //Condition to decrypt odd characters  **else if(c == alphabets[j] && i%2!=0){**  **int temp = (j + key) % 26;**  **decrypted[i] = alphabets[temp];**  **found = 1;**  **break;**  **}**  **}**    //Condition to handle SPACE  **if (!found) {**  **decrypted[i] = c;**  **}**  **}**  //Printing the decrypted text  **decrypted[length = '\0'];**  **printf("%s\n", decrypted);**  **}**  **void main(){**  **char alpha[26] = {'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'};**  **char message[100];**  **printf("Enter your message : ");**  **gets(message);**      //Converting the text into upper case for uniformity  **strupr(message);**  **printf("Converting into Upper Case : \0");**  **printf(message);**  **printf("\nThe Ciphered text is : \0");**  **encrypt(message,alpha,encrypted);**  **printf("The decrypted text is : \0");**  **decrypt(encrypted,alpha,decrypted);**  **}** |

**Output Screenshot:**

****

**Comparative analysis of original and revised approaches:**

Normally, as Caeser Cipher is simple due to possibility of only 25 maximum shifts an attacker can guess or perform a brute force attack where each possibility can be tested and decode the encrypted message but with the revised approach it would be difficult to decode it. Reasons being:

1. Repetition of corresponding shift value.

Example- “RACHIT” from the revised approach has “M” corresponding to “R” but it again has “M” as encrypted value when encountered with “H”.

1. Difference in the shift value of the same character.

Example- “H” in RACHIT has “M” as it’s shift value but “H” in SHAH has “C” as it’s shift value therefore there could be different shift value of a same character

The above points make the decryption of this revised Caeser Cipher text difficult. It’s not the strongest of the encryption but it’s better than the original Caeser Cipher encryption.

**Conclusion:**

Even if the attacker tries to guess the key and tries to decrypt the message it could lead to an unsuccessful attempt.

Example: I AM RACHIT SHAH.

Encrypted message: N FH MFXMDY XCFC.  
Attacker’s assumed key = 5.

Decrypted message of original approach: I AM RACHIT SHAH.  
Decrypted message of revised approach: I AC HASHYT SXAX.

Thus, the revised approach is more secure and better than the original (traditional) Caeser Cipher approach.

**APPLICATIONS:**

The Caesar cipher is used in several areas:

* Education: Introduces students to basic encryption concepts and used in coding practice for beginners.
* Puzzles and Games: Common in escape rooms and cryptographic games.
* Basic Obfuscation: Simple text masking for low-security needs.
* Historical Study: Explored in ancient cryptography research.
* Security Demonstrations: Shows the need for stronger encryption.
* Historical Reenactments: Adds authenticity to simulations of ancient times.