**1)CAESAR CIPHER:**

#include <stdio.h>

#include <ctype.h>

int main() {

char text[500], ch;

int key;

printf("Enter a message: ");

scanf("%s", text);

printf("Enter the key value: ");

scanf("%d", &key);

for (int i = 0; text[i] != '\0'; i++) {

ch = text[i];

if (isalnum(ch)) {

ch = (ch - 'a' + key) % 26 + 'a';

}

text[i] = ch;

}

printf("Encrypted message is: %s", text);

for (int i = 0; text[i] != '\0'; i++) {

ch = text[i];

if (isalnum(ch)) {

ch = (ch - 'a' - key + 26) % 26 + 'a';

}

text[i] = ch;

}

printf("\nDecrypted message is: %s", text);

return 0;

}

**2)PLAY FAIR:**

#include <stdio.h>

#include <string.h>

void prepareKeyMatrix(char key[], char keyMatrix[5][5]) {

int used[26] = {0};

int index = 0;

for (int i = 0; key[i] != '\0'; i++) {

if (key[i] != 'j' && !used[key[i] - 'a']) {

used[key[i] - 'a'] = 1;

keyMatrix[index / 5][index % 5] = key[i];

index++;

}

}

for (int i = 0; i < 26; i++) {

if (i != 'j' - 'a' && !used[i]) {

keyMatrix[index / 5][index % 5] = 'a' + i;

index++;

}

}

}

void encryptPlayfairCipher(char input[], char keyMatrix[5][5]) {

for (int i = 0; input[i] != '\0'; i += 2) {

char a = input[i];

char b = input[i + 1];

int row1, col1, row2, col2;

for (int j = 0; j < 5; j++) {

for (int k = 0; k < 5; k++) {

if (keyMatrix[j][k] == a) {

row1 = j;

col1 = k;

}

if (keyMatrix[j][k] == b) {

row2 = j;

col2 = k;

}

}

}

if (row1 == row2) {

printf("%c%c", keyMatrix[row1][(col1 + 1) % 5], keyMatrix[row2][(col2 + 1) % 5]);

} else if (col1 == col2) {

printf("%c%c", keyMatrix[(row1 + 1) % 5][col1], keyMatrix[(row2 + 1) % 5][col2]);

} else {

printf("%c%c", keyMatrix[row1][col2], keyMatrix[row2][col1]);

}

}

}

int main() {

char key[100], input[100];

char keyMatrix[5][5];

printf("Enter the key (without spaces and 'j' replaced with 'i'): ");

scanf("%s", key);

printf("Enter the input text (without spaces and 'j' replaced with 'i'): ");

scanf("%s", input);

prepareKeyMatrix(key, keyMatrix);

printf("Encrypted text: ");

encryptPlayfairCipher(input, keyMatrix);

return 0;

}

**3)DIFFE HELLMAN:**

#include<stdio.h>

#include<string.h>

#include<math.h>

int main(){

int a,q,xa,xb,ya,yb,x,y,ka,kb,m,n;

printf("Enter the value of a : ");

scanf("%d",&a);

printf("Enter the value of q : ");

scanf("%d",&q);

printf("Enter the value of xa : ");

scanf("%d",&xa);

printf("Enter the value of xb : ");

scanf("%d",&xb);

x=pow(a,xa);

ya=x%q;

y=pow(a,xb);

yb=y%q;

m=pow(yb,xa);

ka=m%q;

n=pow(ya,xb);

kb=n%q;

printf("Secret key of user A : %d ", ka);

printf("\nSecret key of user B : %d ", kb);

}

**4)RSA:**

#include<stdio.h>

#include<string.h>

#include<math.h>

int main(){

int p,q,m,n,dn,e,c,de,x,y;

printf("Enter the value of p : ");

scanf("%d",&p);

printf("Enter the value of q : ");

scanf("%d",&q);

printf("Enter the value of m : ");

scanf("%d",&m);

printf("Enter the value of e : ");

scanf("%d",&e);

n=p\*q;

dn=(p-1)\*(q-1);

int d;

for(int i=1;i<dn;i++){

if(((e%dn)\*(i%dn))%dn==1){

d=i;

break;

}

}

x=pow(m,e);

c=x%n;

y=pow(c,d);

de=y%n;

printf("Encrypted text : %d ", c);

printf("\nDecrypted text : %d ", de);

}

**5)MONOALPHABETIC:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

void monoalphabeticCipher(char \*text, const char \*cipherAlphabet) {

for (; \*text; text++) {

if (isalpha(\*text)) {

int index = toupper(\*text) - 'A';

if (index >= 0 && index < 26) {

\*text = islower(\*text) ? tolower(cipherAlphabet[index]) : cipherAlphabet[index];

}

}

}

}

int main() {

char text[100], cipherAlphabet[27];

printf("Enter a string: ");

fgets(text, sizeof(text), stdin);

text[strcspn(text, "\n")] = '\0';

printf("Enter the cipher alphabet (26 unique uppercase letters): ");

fgets(cipherAlphabet, sizeof(cipherAlphabet), stdin);

cipherAlphabet[strcspn(cipherAlphabet, "\n")] = '\0';

if (strlen(cipherAlphabet) != 26) {

printf("Invalid cipher alphabet. Please enter exactly 26 unique uppercase letters.\n");

return 1;

}

monoalphabeticCipher(text, cipherAlphabet);

printf("Encrypted text: %s\n", text);

return 0;

}

**ii)** #include<stdio.h>

#include<string.h>

#include<ctype.h>

void monoalphabetic(char\*text,const char\*cipheralphabet){

for(int i=0;text[i]!='\0';i++){

if(isalpha(text[i])){

int index=toupper(text[i])-'A';

if(islower(text[i])){

text[i]=tolower(cipheralphabet[index]);

}else{

text[i]=cipheralphabet[index];

}

}

}

}

int main(){

char text[100],cipheralphabet[27];

printf("enter the plaintext:");

fgets(text,sizeof(text),stdin);

text[strcspn(text,"\n")]='\0';

printf("enter cipheralphabet:");

fgets(cipheralphabet,sizeof(cipheralphabet),stdin);

cipheralphabet[strcspn(cipheralphabet,"\n")]='\0';

if(strlen(cipheralphabet)!=26){

printf("INVALID");

return 1;

}

monoalphabetic(text,cipheralphabet);

printf("encryption:%s",text);

}

**6)DES:**  
#include <stdio.h>

#include <stdint.h>

void des\_encrypt(uint64\_t plainText, uint64\_t key, uint64\_t \*cipherText);

void print\_binary(uint64\_t num);

int main() {

uint64\_t plainText, key, cipherText;

printf("Enter the 64-bit plaintext (in hexadecimal): ");

scanf("%llx", &plainText);

printf("Enter the 64-bit encryption key (in hexadecimal): ");

scanf("%llx", &key);

des\_encrypt(plainText, key, &cipherText);

printf("\nPlaintext: ");

print\_binary(plainText);

printf("\nKey: ");

print\_binary(key);

printf("\nCiphertext: ");

print\_binary(cipherText);

return 0;

}

void des\_encrypt(uint64\_t plainText, uint64\_t key, uint64\_t \*cipherText) {

\*cipherText = plainText;

}

void print\_binary(uint64\_t num) {

for (int i = 63; i >= 0; i--) {

uint64\_t bit = (num >> i) & 1;

printf("%lu", bit);

if (i % 8 == 0) {

printf(" ");

}

}

printf("\n");

}

**OUTPUT:**

Enter the 64-bit plaintext (in hexadecimal): 0123456789ABCDEF

Enter the 64-bit encryption key (in hexadecimal): 133457799BBCDFF1

Plaintext: 00000001 00100011 01000101 01100111 10001001 10101111 11001101 11101111

Key: 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

Ciphertext: 00000001 00100011 01000101 01100111 10001001 10101111 11001101 11101111

**7)HILL CIPHER:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int mod(int a, int m) {

int result = a % m;

if (result < 0) {

result += m;

}

return result;

}

void hillCipherEncrypt(int keyMatrix[3][3], char\* input, int len) {

int encrypted[len];

printf("Plaintext: %s\n", input);

for (int i = 0; i < len; i += 3) {

for (int j = 0; j < 3; j++) {

encrypted[i + j] = 0;

for (int k = 0; k < 3; k++) {

encrypted[i + j] += keyMatrix[j][k] \* (input[i + k] - 'A');

}

encrypted[i + j] = mod(encrypted[i + j], 26);

}

}

printf("Ciphertext: ");

for (int i = 0; i < len; i++) {

printf("%c", encrypted[i] + 'A');

}

printf("\n");

}

int main() {

int keyMatrix[3][3];

char input[100];

int len;

printf("Enter the 3x3 key matrix:\n");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

scanf("%d", &keyMatrix[i][j]);

}

}

printf("Enter the plaintext (in uppercase): ");

scanf("%s", input);

len = strlen(input);

while (len % 3 != 0) {

input[len] = 'X';

len++;

}

hillCipherEncrypt(keyMatrix, input, len);

return 0;

}

**ii)** **#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**int mod(int a, int m) {**

**return (a % m + m) % m;**

**}**

**void hillCipherEncrypt(int keyMatrix[3][3], char\* input, int len) {**

**int encrypted[len];**

**for (int i = 0; i < len; i += 3) {**

**for (int j = 0; j < 3; j++) {**

**encrypted[i + j] = 0;**

**for (int k = 0; k < 3; k++) {**

**encrypted[i + j] += keyMatrix[j][k] \* (input[i + k] - 'A');**

**}**

**encrypted[i + j] = mod(encrypted[i + j], 26);**

**}**

**}**

**printf("Ciphertext: ");**

**for (int i = 0; i < len; i++) {**

**printf("%c", encrypted[i] + 'A');**

**}**

**printf("\n");**

**}**

**int main() {**

**int keyMatrix[3][3];**

**char input[100];**

**printf("Enter the 3x3 key matrix:\n");**

**for (int i = 0; i < 3; i++) {**

**for (int j = 0; j < 3; j++) {**

**scanf("%d", &keyMatrix[i][j]);**

**}**

**}**

**printf("Enter the plaintext (in uppercase): ");**

**scanf("%s", input);**

**int len = strlen(input);**

**while (len % 3 != 0) {**

**input[len++] = 'X';**

**}**

**input[len] = '\0';**

**hillCipherEncrypt(keyMatrix, input, len);**

**return 0;**

**}**

**OUTPUT:**  
Enter the 3×3 key matrix:

17 21 2

17 18 2

5 21 19

Enter the plaintext (in uppercase): PAY

Plaintext: PAY

Ciphertext: RRL