**CRYTOGRAPHY ANSWERS:**

**CAESER CIPHER:**

//ques-1

#include <stdio.h>

#include <ctype.h>

int main() {

char text[100];

int shift, i = 0;

char ch;

printf("Enter a message to encrypt: ");

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

printf("Enter the shift value (1-25): ");

scanf("%d", &shift);

if (shift < 0) {

shift = 26 + (shift % 26);

}

shift = shift % 26;

while (text[i] != '\0') {

ch = text[i];

if (isupper(ch)) {

ch = (ch - 'A' + shift) % 26 + 'A';

text[i] = ch;

}

else if (islower(ch)) {

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

text[i] = ch;

}

i++;

}

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

return 0;

}

**MONOALPHABETIC CIPHER:**

#include <stdio.h>

#include <string.h>

int main() {

char plaintext[100], ciphertext[100];

char alphabet[] = "abcdefghijklmnopqrstuvwxyz";

char key[] = "qwertyuiopasdfghjklzxcvbnm"; // The substitution key

printf("Enter the plaintext (lowercase letters only): ");

gets(plaintext); // Take input of plaintext

int length = strlen(plaintext);

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

if (plaintext[i] >= 'a' && plaintext[i] <= 'z') {

ciphertext[i] = key[plaintext[i] - 'a']; // Substituting the letter

} else {

ciphertext[i] = plaintext[i]; // If not a lowercase letter, leave it as it is

}

}

ciphertext[length] = '\0'; // Null-terminate the ciphertext

printf("Ciphertext: %s\n", ciphertext);

return 0;

}

**PLAYFAIR CIPHER:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 5

int main() {

char key[100], plaintext[100], ciphertext[100];

char matrix[SIZE][SIZE];

int used[26] = {0};

int i, j, k = 0, row1, col1, row2, col2;

printf("Enter the key(without duplicates): ");

fgets(key, sizeof(key), stdin);

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

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

if (key[i] == 'J') key[i] = 'I';

key[i] = toupper(key[i]);

}

for (i = 0; i < strlen(key); i++) {

if (!used[key[i] - 'A'] && isalpha(key[i])) {

matrix[k / SIZE][k % SIZE] = key[i];

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

k++;

}

}

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

if (i != ('J' - 'A') && !used[i]) {

matrix[k / SIZE][k % SIZE] = 'A' + i;

used[i] = 1;

k++;

}

}

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

fgets(plaintext, sizeof(plaintext), stdin);

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

int len = strlen(plaintext);

for (i = 0; i < len; i += 2) {

if (plaintext[i] == plaintext[i+1]) {

for (j = len; j > i+1; j--) {

plaintext[j] = plaintext[j-1];

}

plaintext[i+1] = 'X';

len++;

}

if (len % 2 != 0) {

plaintext[len] = 'X';

len++;

plaintext[len] = '\0';

}

for (row1 = 0; row1 < SIZE; row1++) {

for (col1 = 0; col1 < SIZE; col1++) {

if (matrix[row1][col1] == plaintext[i]) break;

}

if (col1 < SIZE) break;

}

for (row2 = 0; row2 < SIZE; row2++) {

for (col2 = 0; col2 < SIZE; col2++) {

if (matrix[row2][col2] == plaintext[i+1]) break;

}

if (col2 < SIZE) break;

}

if (row1 == row2) {

ciphertext[i] = matrix[row1][(col1 + 1) % SIZE];

ciphertext[i+1] = matrix[row2][(col2 + 1) % SIZE];

} else if (col1 == col2) {

ciphertext[i] = matrix[(row1 + 1) % SIZE][col1];

ciphertext[i+1] = matrix[(row2 + 1) % SIZE][col2];

} else {

ciphertext[i] = matrix[row1][col2];

ciphertext[i+1] = matrix[row2][col1];

}

}

ciphertext[len] = '\0';

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

return 0;

}

**POLY ALPHABETIC CIPHER:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char plaintext[100], key[100], ciphertext[100];

int i, j = 0, keyLength, plaintextLength;

printf("Enter the plaintext (uppercase letters only): ");

fgets(plaintext, sizeof(plaintext), stdin);

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

plaintextLength = strlen(plaintext);

printf("Enter the key (uppercase letters only): ");

fgets(key, sizeof(key), stdin);

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

keyLength = strlen(key);

for (i = 0; i < plaintextLength; i++) {

if (isalpha(plaintext[i])) {

int shift = key[j % keyLength] - 'A';

if (isupper(plaintext[i])) {

ciphertext[i] = ((plaintext[i] - 'A' + shift) % 26) + 'A';

} else {

ciphertext[i] = plaintext[i];

}

j++;

} else {

ciphertext[i] = plaintext[i];

}

}

ciphertext[plaintextLength] = '\0';

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

return 0;

}

**AFFINE CAESER CIPHER:**

#include <stdio.h>

#include <ctype.h>

#include<string.h>

int modInverse(int a, int m) {

int m0 = m, t, q;

int x0 = 0, x1 = 1;

if (m == 1)

return 0;

while (a > 1) {

q = a / m;

t = m;

m = a % m, a = t;

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

if (x1 < 0)

x1 += m0;

return x1;

}

void encrypt(char plaintext[], int a, int b) {

int i = 0;

char ciphertext[100];

while (plaintext[i] != '\0') {

if (isalpha(plaintext[i])) {

char letter = toupper(plaintext[i]);

ciphertext[i] = ((a \* (letter - 'A') + b) % 26) + 'A';

} else {

ciphertext[i] = plaintext[i];

}

i++;

}

ciphertext[i] = '\0';

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

}

void decrypt(char ciphertext[], int a, int b) {

int i = 0;

char decrypted[100];

int a\_inv = modInverse(a, 26);

while (ciphertext[i] != '\0') {

if (isalpha(ciphertext[i])) {

char letter = toupper(ciphertext[i]);

decrypted[i] = (a\_inv \* ((letter - 'A') - b + 26) % 26) + 'A';

} else {

decrypted[i] = ciphertext[i];

}

i++;

}

decrypted[i] = '\0';

printf("Decrypted message: %s\n", decrypted);

}

int main() {

char plaintext[100], ciphertext[100];

int a, b;

printf("Enter the plaintext (uppercase letters only): ");

fgets(plaintext, sizeof(plaintext), stdin);

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

printf("Enter the value of a (must be coprime with 26): ");

scanf("%d", &a);

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

scanf("%d", &b);

encrypt(plaintext, a, b);

printf("Enter the ciphertext to decrypt (uppercase letters only): ");

scanf("%s", ciphertext);

decrypt(ciphertext, a, b);

return 0;

}

**POWER OF 2:**

#include <stdio.h>

#include <math.h>

int main() {

double log2\_factorial = 0.0;

for (int i = 1; i <= 25; i++) {

log2\_factorial += log2(i);

}

printf("log2(25!) = %.2lf\n", log2\_factorial);

printf("Approximate power of 2: 2^%.2lf\n", log2\_factorial);

return 0;

}

**PLAYFAIR CODE-MATRIX;**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

char keyTable[5][5] = {

{'M', 'F', 'H', 'I', 'K'},

{'U', 'N', 'O', 'P', 'Q'},

{'Z', 'V', 'W', 'X', 'Y'},

{'E', 'L', 'A', 'R', 'G'},

{'D', 'S', 'T', 'B', 'C'}

};

void prepareText(char input[], char preparedText[]);

void findPosition(char letter, int \*row, int \*col);

void encryptPlayfair(char digraph[], char encryptedDigraph[]);

void playfairEncrypt(char plaintext[], char encryptedText[]);

int main() {

char plaintext[] = "Must see you over Cadogan West. Coming at once.";

char encryptedText[256];

playfairEncrypt(plaintext, encryptedText);

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

return 0;

}

void prepareText(char input[], char preparedText[]) {

int len = strlen(input);

int index = 0;

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

if (isalpha(input[i])) {

char ch = toupper(input[i]);

if (ch == 'J') {

ch = 'I';

}

preparedText[index++] = ch;

}

}

preparedText[index] = '\0';

}

void findPosition(char letter, int \*row, int \*col) {

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

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

if (keyTable[i][j] == letter) {

\*row = i;

\*col = j;

return;

}

}

}

}

void encryptPlayfair(char digraph[], char encryptedDigraph[]) {

int row1, col1, row2, col2;

findPosition(digraph[0], &row1, &col1);

findPosition(digraph[1], &row2, &col2);

if (row1 == row2) {

encryptedDigraph[0] = keyTable[row1][(col1 + 1) % 5];

encryptedDigraph[1] = keyTable[row2][(col2 + 1) % 5];

} else if (col1 == col2) {

encryptedDigraph[0] = keyTable[(row1 + 1) % 5][col1];

encryptedDigraph[1] = keyTable[(row2 + 1) % 5][col2];

} else {

encryptedDigraph[0] = keyTable[row1][col2];

encryptedDigraph[1] = keyTable[row2][col1];

}

}

void playfairEncrypt(char plaintext[], char encryptedText[]) {

char preparedText[256];

char digraph[3], encryptedDigraph[3];

int len, index = 0;

prepareText(plaintext, preparedText);

len = strlen(preparedText);

for (int i = 0; i < len; i += 2) {

digraph[0] = preparedText[i];

if (i + 1 < len) {

digraph[1] = preparedText[i + 1];

} else {

digraph[1] = 'X';

}

digraph[2] = '\0';

encryptPlayfair(digraph, encryptedDigraph);

encryptedText[index++] = encryptedDigraph[0];

encryptedText[index++] = encryptedDigraph[1];

}

encryptedText[index] = '\0';

}

**PT-109 AMERICAN PATROL:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void generateKeyTable(char key[], char keyTable[5][5]);

void prepareText(char str[], char preparedText[]);

void decryptPlayfair(char digraph[], char keyTable[5][5], char decryptedDigraph[]);

void findPosition(char letter, char keyTable[5][5], int \*row, int \*col);

void playfairDecrypt(char ciphertext[], char key[], char decryptedText[]);

int main() {

char ciphertext[] = "KXJEYUREBEZWEHEWRYTUHEYFSKREHEGOYFIWTTTUOLKSYCAJPOBOTEIZONTXBYBNTGONECUZWRGDSONSXBOUYWRHEBAAHYUSEDQ";

char key[] = "JFK";

char decryptedText[256];

playfairDecrypt(ciphertext, key, decryptedText);

printf("Decrypted message: %s\n", decryptedText);

return 0;

}

void generateKeyTable(char key[], char keyTable[5][5]) {

int used[26] = {0};

int row = 0, col = 0;

used['J' - 'A'] = 1;

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

if (!used[toupper(key[i]) - 'A']) {

keyTable[row][col++] = toupper(key[i]);

used[toupper(key[i]) - 'A'] = 1;

if (col == 5) {

row++;

col = 0;

}

}

}

for (char ch = 'A'; ch <= 'Z'; ch++) {

if (!used[ch - 'A']) {

keyTable[row][col++] = ch;

used[ch - 'A'] = 1;

if (col == 5) {

row++;

col = 0;

}

}

}

}

void prepareText(char str[], char preparedText[]) {

int len = strlen(str), index = 0;

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

if (isalpha(str[i])) {

preparedText[index++] = toupper(str[i]);

}

}

preparedText[index] = '\0';

}

void decryptPlayfair(char digraph[], char keyTable[5][5], char decryptedDigraph[]) {

int row1, col1, row2, col2;

findPosition(digraph[0], keyTable, &row1, &col1);

findPosition(digraph[1], keyTable, &row2, &col2);

if (row1 == row2) {

decryptedDigraph[0] = keyTable[row1][(col1 + 4) % 5];

decryptedDigraph[1] = keyTable[row2][(col2 + 4) % 5];

} else if (col1 == col2) {

decryptedDigraph[0] = keyTable[(row1 + 4) % 5][col1];

decryptedDigraph[1] = keyTable[(row2 + 4) % 5][col2];

} else {

decryptedDigraph[0] = keyTable[row1][col2];

decryptedDigraph[1] = keyTable[row2][col1];

}

}

void findPosition(char letter, char keyTable[5][5], int \*row, int \*col) {

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

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

if (keyTable[i][j] == letter) {

\*row = i;

\*col = j;

return;

}

}

}

}

void playfairDecrypt(char ciphertext[], char key[], char decryptedText[]) {

char keyTable[5][5];

char preparedCiphertext[256], digraph[3], decryptedDigraph[3];

int len, index = 0;

generateKeyTable(key, keyTable);

prepareText(ciphertext, preparedCiphertext);

len = strlen(preparedCiphertext);

for (int i = 0; i < len; i += 2) {

digraph[0] = preparedCiphertext[i];

digraph[1] = preparedCiphertext[i + 1];

digraph[2] = '\0';

decryptPlayfair(digraph, keyTable, decryptedDigraph);

decryptedText[index++] = decryptedDigraph[0];

decryptedText[index++] = decryptedDigraph[1];

}

decryptedText[index] = '\0';

}

**SUBSTITUTION CIPHER:**

#include <stdio.h>

#include <string.h>

int main() {

char plaintext[100], ciphertext[100];

char key[27] = "QWERTYUIOPLKJHGFDSAZXCVBNM";

char alphabet[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int i, j;

printf("Enter a message to encrypt(uppercase): ");

fgets(plaintext, sizeof(plaintext), stdin);

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

if (plaintext[i] >= 'A' && plaintext[i] <= 'Z') {

for (j = 0; j < 26; j++) {

if (plaintext[i] == alphabet[j]) {

ciphertext[i] = key[j];

break;

}

}

} else {

ciphertext[i] = plaintext[i];

}

}

ciphertext[i] = '\0';

printf("Encrypted message: %s", ciphertext);

return 0;

}