**CSA5127 -CRYPTOGRAPHY AND NETWORK SECURITY FOR BANKING SECTOR**

**LAB ACTIVITY**

**PARAMESWARI K**

**192372266**

**1. CAESAR CIPHER TECHNIQUE**

#include<stdio.h>

#include<ctype.h>

void encrypt(const char\* input){

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

char ch=toupper(input[i]);

if(ch>='A'&&ch<='Z'){

printf(" %d",ch-64);

}

else if(ch==' '){

printf(" ");

}

}

printf("\n");

}

int main(){

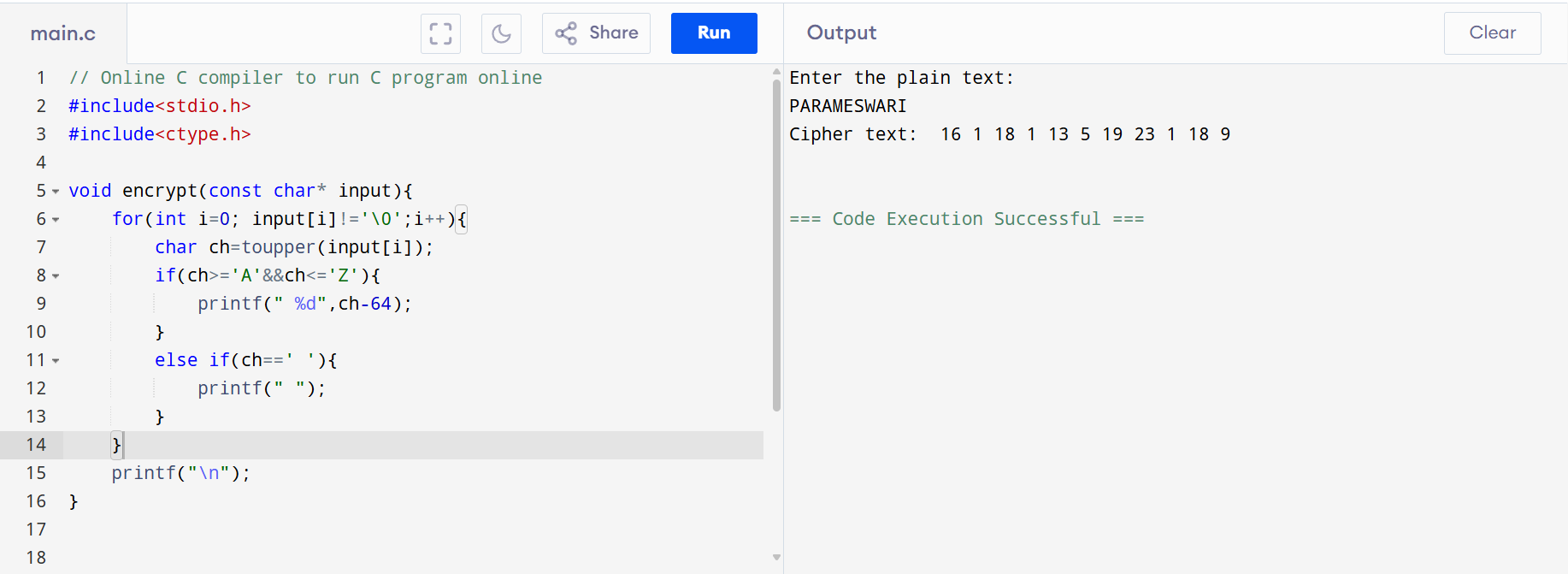
char msg[100];

printf("Enter the plain text:\n");

fgets(msg,sizeof(msg),stdin);

printf("Cipher text: ");

encrypt(msg);

****

**2.MONOALPHABETIC TECHNIQUE**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

// Substitution key (26 characters for A-Z)

char key[26] = {

'Q','W','E','R','T','Y','U','I','O','P',

'A','S','D','F','G','H','J','K','L','Z',

'X','C','V','B','N','M'

};

void encrypt(char\* text) {

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

if (isupper(text[i]))

text[i] = key[text[i] - 'A'];

else if (islower(text[i]))

text[i] = tolower(key[text[i] - 'a']);

}

}

void decrypt(char\* text) {

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

if (isalpha(text[i])) {

char ch = toupper(text[i]);

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

if (key[j] == ch) {

text[i] = isupper(text[i]) ? ('A' + j) : tolower('A' + j);

break;

}

}

}

}

}

int main() {

char text[100];

int choice;

printf("Enter message: ");

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

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

printf("Choose option:\n1. Encrypt\n2. Decrypt\nEnter choice: ");

scanf("%d", &choice);

if (choice == 1) {

encrypt(text);

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

} else if (choice == 2) {

decrypt(text);

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

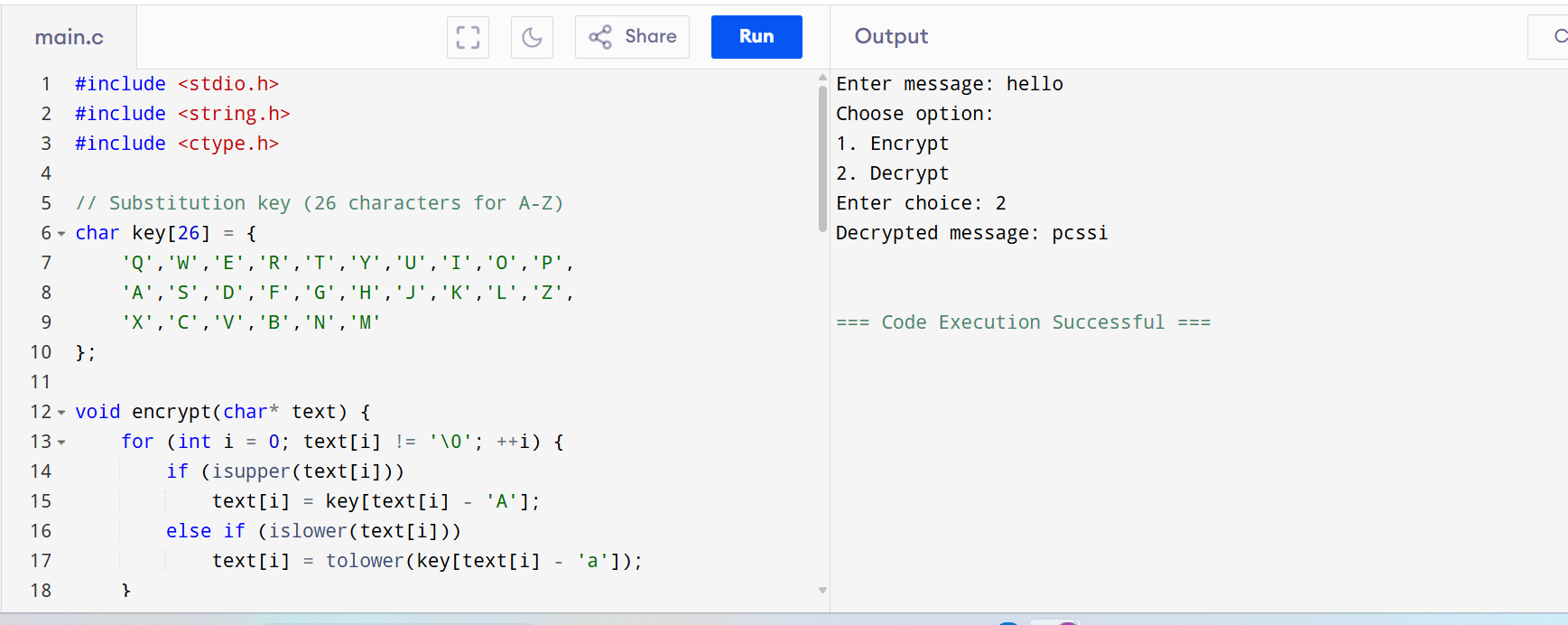
} else {

printf("Invalid choice.\n");

}

return 0;

}



**3.POLYALPHABETIC SUBSTITUTION TECHNIQUE**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char text[100], key[100], newKey[100];

int choice, i, j = 0;

printf("Enter the text: ");

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

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

printf("Enter the key: ");

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

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

// Convert key to uppercase

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

if (isalpha(key[i]))

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

}

// Generate repeated key

int textLen = strlen(text);

int keyLen = strlen(key);

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

if (isalpha(text[i])) {

newKey[i] = key[j % keyLen];

j++;

} else {

newKey[i] = text[i]; // maintain spaces/symbols

}

}

newKey[i] = '\0';

printf("Choose:\n1. Encrypt\n2. Decrypt\n");

scanf("%d", &choice);

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

if (isalpha(text[i])) {

char base = isupper(text[i]) ? 'A' : 'a';

char keyBase = toupper(newKey[i]) - 'A';

if (choice == 1) {

text[i] = ((toupper(text[i]) - 'A' + keyBase) % 26) + base;

} else if (choice == 2) {

text[i] = ((toupper(text[i]) - 'A' - keyBase + 26) % 26) + base;

}

}

}

if (choice == 1)

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

else if (choice == 2)

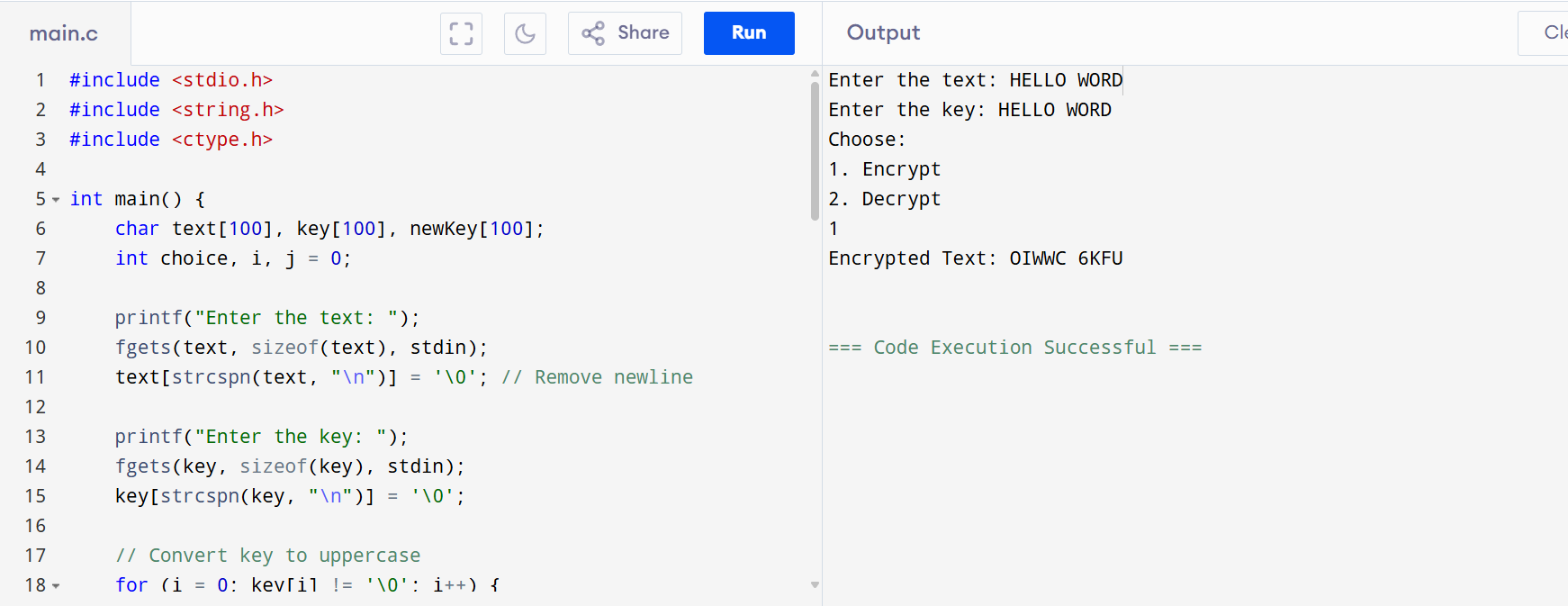
printf("Decrypted Text: %s\n", text);

else

printf("Invalid choice\n");

return 0;

}



**4. HILL CIPHER**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

// Convert character to number (A=0 to Z=25)

int charToInt(char ch) {

return toupper(ch) - 'A';

}

// Convert number to character (0=A to 25=Z)

char intToChar(int num) {

return num + 'A';

}

int main() {

char plaintext[100], ciphertext[100];

int key[2][2];

int pt[2], ct[2];

int i, j = 0;

printf("Enter 2x2 key matrix (integers mod 26):\n");

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

for (j = 0; j < 2; j++)

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

getchar(); // clear newline

printf("Enter plaintext: ");

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

// Remove non-letters and convert to uppercase

char cleanText[100];

j = 0;

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

if (isalpha(plaintext[i])) {

cleanText[j++] = toupper(plaintext[i]);

}

}

cleanText[j] = '\0';

// Pad with 'X' if length is odd

if (strlen(cleanText) % 2 != 0) {

cleanText[strlen(cleanText)] = 'X';

cleanText[strlen(cleanText) + 1] = '\0';

}

// Encrypt

int len = strlen(cleanText);

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

pt[0] = charToInt(cleanText[i]);

pt[1] = charToInt(cleanText[i + 1]);

ct[0] = (key[0][0] \* pt[0] + key[0][1] \* pt[1]) % 26;

ct[1] = (key[1][0] \* pt[0] + key[1][1] \* pt[1]) % 26;

ciphertext[i] = intToChar(ct[0]);

ciphertext[i + 1] = intToChar(ct[1]);

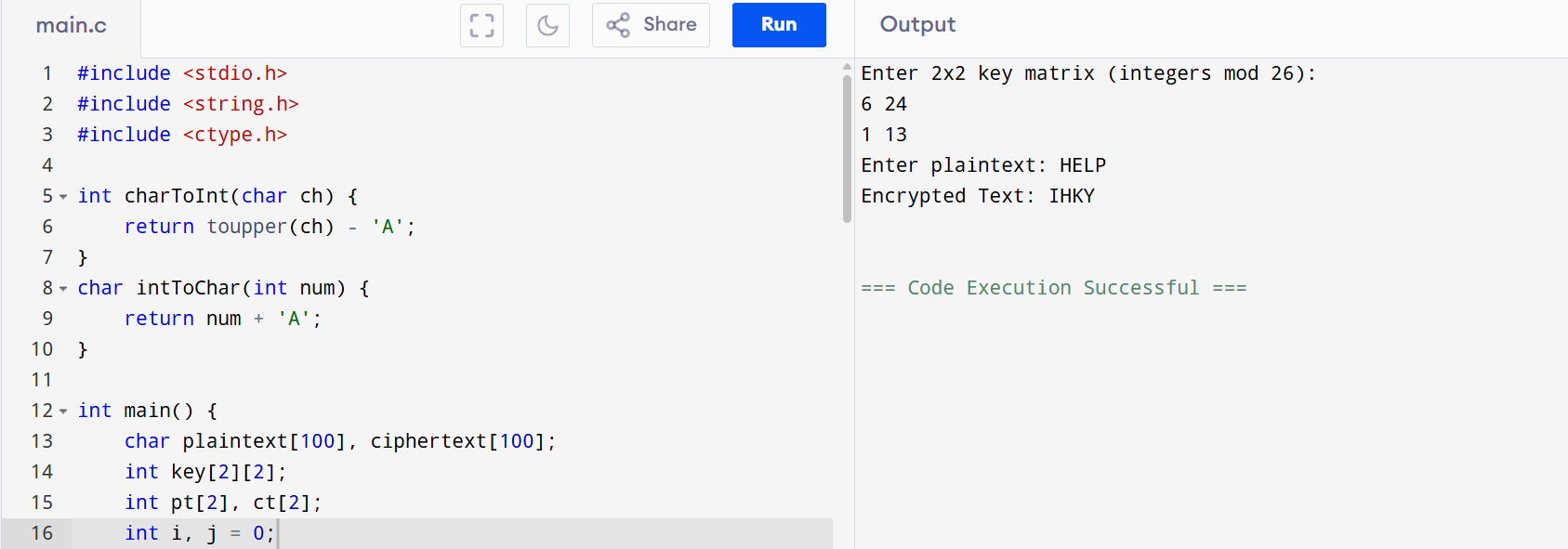
}

ciphertext[len] = '\0';

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

return 0;

}



**5. COLUMNAR TRANSPOSITION TECHNIQUE**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#define MAX 100

// Structure to hold character and its position in key

typedef struct {

char ch;

int pos;

} KeyChar;

// Compare function for qsort

int compare(const void \*a, const void \*b) {

KeyChar \*k1 = (KeyChar \*)a;

KeyChar \*k2 = (KeyChar \*)b;

return k1->ch - k2->ch;

}

void encrypt(char \*plaintext, char \*key) {

int len = strlen(plaintext);

int keyLen = strlen(key);

int row = (len + keyLen - 1) / keyLen;

// Fill matrix with text

char mat[row][keyLen];

int k = 0;

for (int i = 0; i < row; i++)

for (int j = 0; j < keyLen; j++)

mat[i][j] = (k < len) ? plaintext[k++] : 'X'; // pad with 'X'

// Prepare key index structure

KeyChar keyArr[keyLen];

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

keyArr[i].ch = toupper(key[i]);

keyArr[i].pos = i;

}

// Sort the key

qsort(keyArr, keyLen, sizeof(KeyChar), compare);

// Print encrypted text column by column

printf("Encrypted Text: ");

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

int col = keyArr[k].pos;

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

printf("%c", mat[i][col]);

}

}

printf("\n");

}

void decrypt(char \*ciphertext, char \*key) {

int len = strlen(ciphertext);

int keyLen = strlen(key);

int row = (len + keyLen - 1) / keyLen;

char mat[row][keyLen];

// Prepare key with original indexes

KeyChar keyArr[keyLen];

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

keyArr[i].ch = toupper(key[i]);

keyArr[i].pos = i;

}

// Sort key

KeyChar sortedKey[keyLen];

memcpy(sortedKey, keyArr, sizeof(keyArr));

qsort(sortedKey, keyLen, sizeof(KeyChar), compare);

// Fill matrix column-wise based on sorted key

int k = 0;

for (int x = 0; x < keyLen; x++) {

int col = sortedKey[x].pos;

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

mat[i][col] = (k < len) ? ciphertext[k++] : 'X';

}

}

// Read matrix row-wise

printf("Decrypted Text: ");

for (int i = 0; i < row; i++)

for (int j = 0; j < keyLen; j++)

printf("%c", mat[i][j]);

printf("\n");

}

int main() {

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

int choice;

printf("Enter the key: ");

fgets(key, MAX, stdin);

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

printf("Choose:\n1. Encrypt\n2. Decrypt\nChoice: ");

scanf("%d", &choice);

getchar(); // clear newline

if (choice == 1) {

printf("Enter plaintext: ");

fgets(plaintext, MAX, stdin);

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

// Clean text: remove non-letters and convert to uppercase

int j = 0;

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

if (isalpha(plaintext[i]))

plaintext[j++] = toupper(plaintext[i]);

}

plaintext[j] = '\0';

encrypt(plaintext, key);

} else if (choice == 2) {

printf("Enter ciphertext: ");

fgets(ciphertext, MAX, stdin);

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

decrypt(ciphertext, key);

} else {

printf("Invalid choice.\n");

}

return 0;

}

