//CAESER CIPHER

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

void getKeyOrder(char key[], int order[]) {

int len = strlen(key);

char sortedKey[20];

strcpy(sortedKey, key);

for (int i = 0; i < len - 1; i++) {

for (int j = i + 1; j < len; j++) {

if (sortedKey[i] > sortedKey[j]) {

char temp = sortedKey[i];

sortedKey[i] = sortedKey[j];

sortedKey[j] = temp;

}

}

}

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

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

if (key[i] == sortedKey[j] && order[j] == -1) {

order[j] = i;

break;

}

}

}

}

void encrypt(char message[], char key[]) {

int msgLen = strlen(message);

int keyLen = strlen(key);

int rows = (msgLen + keyLen - 1) / keyLen;

char matrix[rows][keyLen];

int k = 0;

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

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

if (k < msgLen)

matrix[i][j] = message[k++];

else

matrix[i][j] = 'X';

}

}

int order[20];

for (int i = 0; i < keyLen; i++) order[i] = -1;

getKeyOrder(key, order);

printf("Encrypted message: ");

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

int actualCol = order[col];

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

printf("%c", matrix[row][actualCol]);

}

}

printf("\n");

}

void decrypt(char cipher[], char key[]) {

int cipherLen = strlen(cipher);

int keyLen = strlen(key);

int rows = (cipherLen + keyLen - 1) / keyLen;

char matrix[rows][keyLen];

int order[20];

for (int i = 0; i < keyLen; i++) order[i] = -1;

getKeyOrder(key, order);

int k = 0;

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

int actualCol = order[col];

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

if (k < cipherLen)

matrix[row][actualCol] = cipher[k++];

}

}

printf("Decrypted message: ");

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

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

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

}

}

printf("\n");

}

int main() {

char message[100], key[20];

int choice;

printf("1. Encrypt\n2. Decrypt\nEnter your choice: ");

scanf("%d", &choice);

getchar();

if (choice == 1) {

printf("Enter the plaintext (no spaces): ");

scanf("%s", message);

printf("Enter the keyword: ");

scanf("%s", key);

encrypt(message, key);

} else if (choice == 2) {

printf("Enter the ciphertext (no spaces): ");

scanf("%s", message);

printf("Enter the keyword: ");

scanf("%s", key);

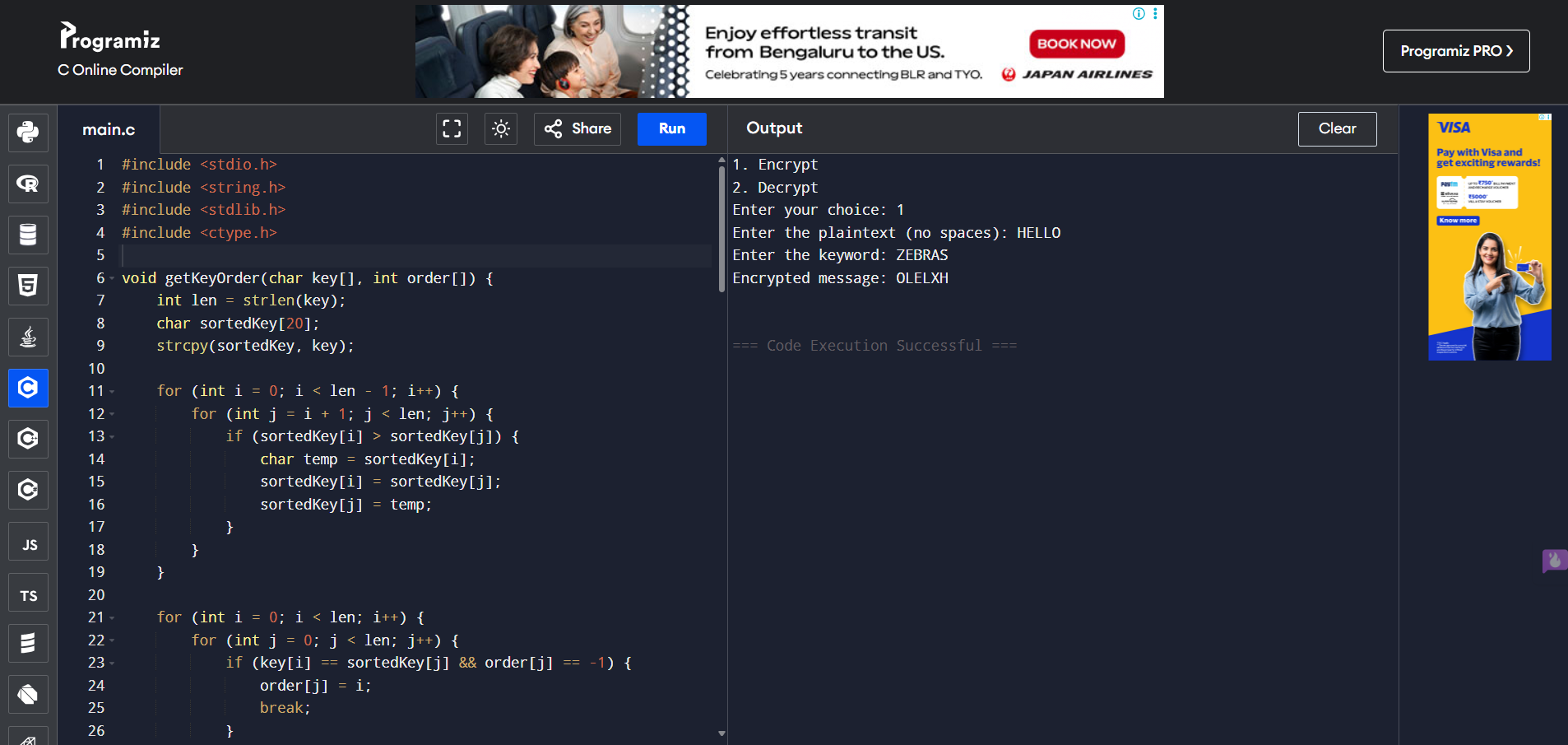
decrypt(message, key);

} else {

printf("Invalid choice.\n");

}

return 0;

}

//COLUMNAR CIPHER

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#define MAX 100

typedef struct {

char ch;

int pos;

} KeyChar;

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;

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';

KeyChar keyArr[keyLen];

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

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

keyArr[i].pos = i;

}

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

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];

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';

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

scanf("%d", &choice);

getchar();

if (choice == 1) {

printf("Enter plaintext: ");

fgets(plaintext, MAX, stdin);

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

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);

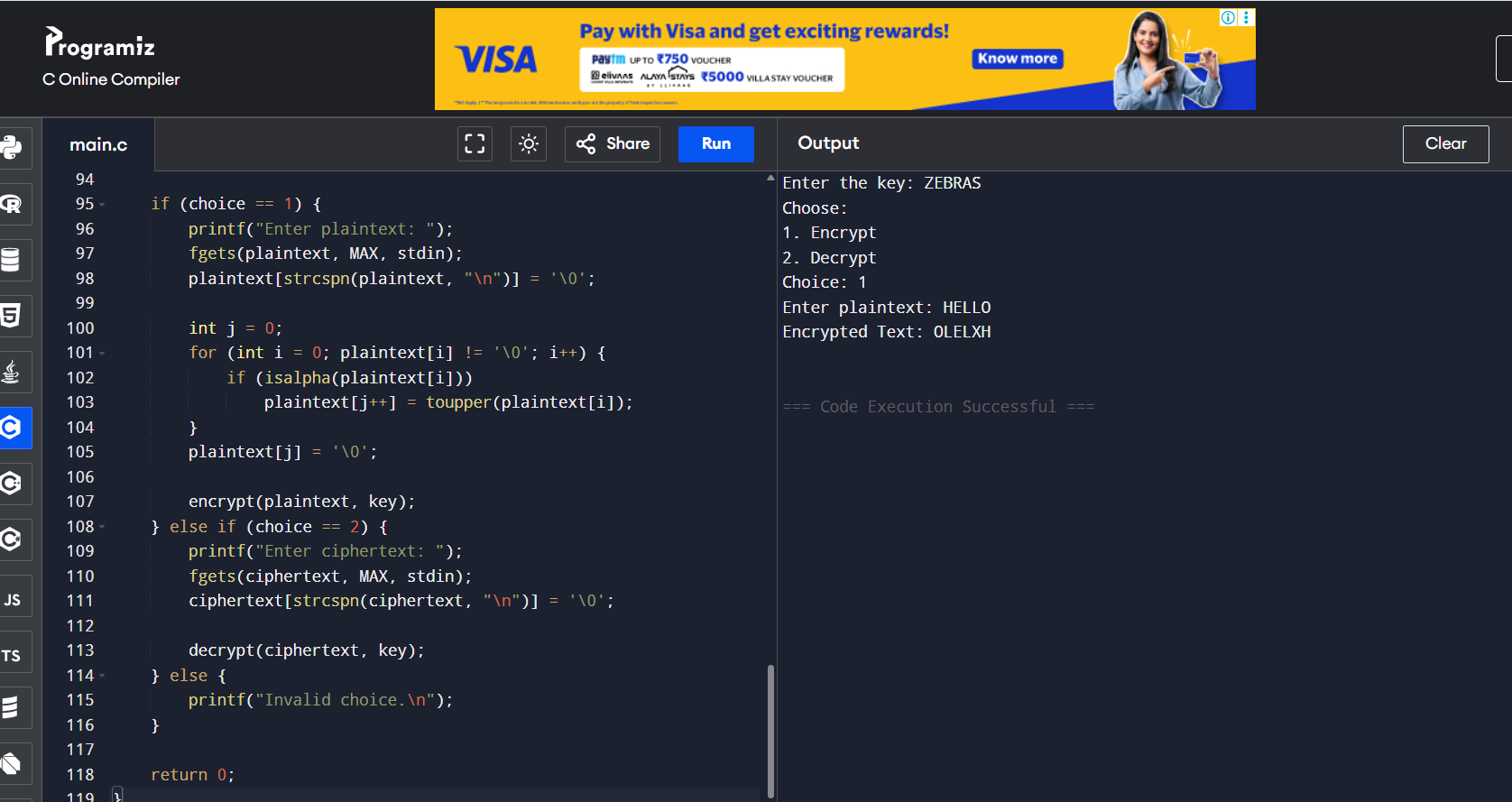
ciphertext[strcspn(ciphertext, "\n")] = '\0';

decrypt(ciphertext, key);

} else { printf("Invalid choice.\n");

}

return 0;

}

//MONOALPHABETIC CIPHER

#include <stdio.h>

#include <string.h>

#include <ctype.h>

char keyMap[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 message[]) {

char encrypted[100];

int i;

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

char ch = toupper(message[i]);

if (isalpha(ch)) {

encrypted[i] = keyMap[ch - 'A'];

} else {

encrypted[i] = ch;

}

}

encrypted[i] = '\0';

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

}

void decrypt(char cipher[]) {

char decrypted[100];

int i;

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

char ch = toupper(cipher[i]);

if (isalpha(ch)) {

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

if (keyMap[j] == ch) {

decrypted[i] = 'A' + j;

break;

}

}

} else {

decrypted[i] = ch;

}

}

decrypted[i] = '\0';

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

}

int main() {

int choice;

char text[100];

printf("1. Encrypt\n2. Decrypt\nEnter your choice: ");

scanf("%d", &choice);

getchar();

printf("Enter the message (no spaces): ");

scanf("%s", text);

if (choice == 1)

encrypt(text);

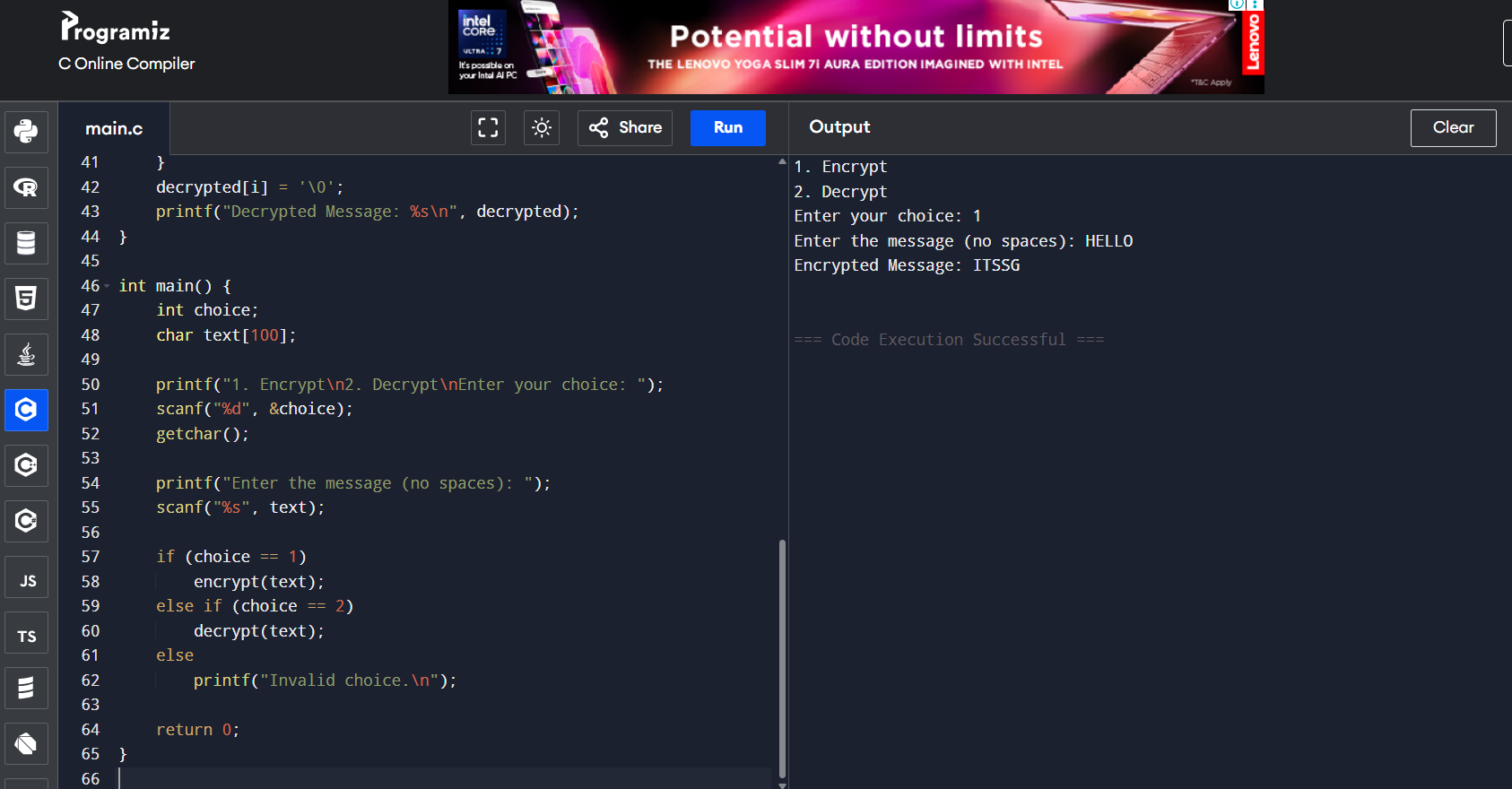
else if (choice == 2)

decrypt(text);

else

printf("Invalid choice.\n");

return 0;

}

//VEGENER CIPHER

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void generateKey(char message[], char key[]) {

int msgLen = strlen(message), keyLen = strlen(key), i;

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

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

}

key[i] = '\0';

}

void encrypt(char message[], char key[], char cipher[]) {

int i;

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

char m = toupper(message[i]);

if (isalpha(m)) {

cipher[i] = ((m - 'A') + (key[i] - 'A')) % 26 + 'A';

} else {

cipher[i] = m;

}

}

cipher[i] = '\0';

}

void decrypt(char cipher[], char key[], char plain[]) {

int i;

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

char c = toupper(cipher[i]);

if (isalpha(c)) {

plain[i] = ((c - key[i] + 26) % 26) + 'A';

} else {

plain[i] = c;

}

}

plain[i] = '\0';

}

int main() {

char message[100], key[100], fullKey[100], cipher[100], plain[100];

int choice;

printf("1. Encrypt\n2. Decrypt\nEnter your choice: ");

scanf("%d", &choice);

getchar();

if (choice == 1) {

printf("Enter message (no spaces): ");

scanf("%s", message);

printf("Enter key (no spaces): ");

scanf("%s", key);

generateKey(message, key);

encrypt(message, key, cipher);

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

} else if (choice == 2) {

printf("Enter cipher text: ");

scanf("%s", cipher);

printf("Enter key: ");

scanf("%s", key);

generateKey(cipher, key);

decrypt(cipher, key, plain);

printf("Decrypted Message: %s\n", plain);

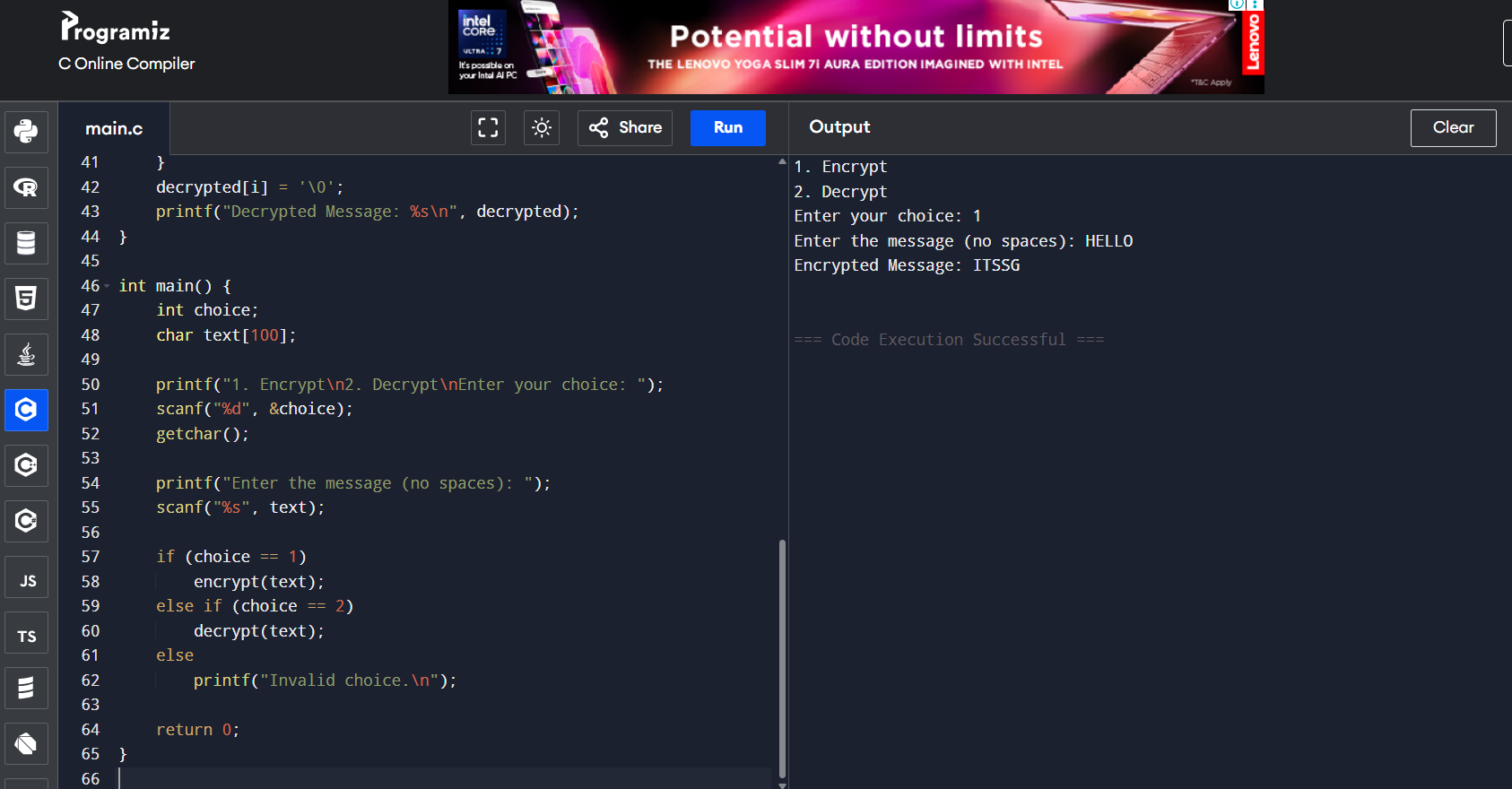
} else {

printf("Invalid choice.\n");

}

return 0;

}



//HILL CIPHER

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 2

int modInverse(int a) {

a = a % 26;

for (int x = 1; x < 26; x++) {

if ((a \* x) % 26 == 1)

return x;

}

return -1;

}

int inverseKey(int key[SIZE][SIZE], int inv[SIZE][SIZE]) {

int det = (key[0][0]\*key[1][1] - key[0][1]\*key[1][0]) % 26;

if (det < 0) det += 26;

int invDet = modInverse(det);

if (invDet == -1) return 0;

inv[0][0] = key[1][1];

inv[0][1] = -key[0][1];

inv[1][0] = -key[1][0];

inv[1][1] = key[0][0];

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

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

inv[i][j] = inv[i][j] \* invDet;

inv[i][j] %= 26;

if (inv[i][j] < 0) inv[i][j] += 26;

}

return 1;

}

int charToNum(char ch) {

return toupper(ch) - 'A';

}

char numToChar(int n) {

return n + 'A';

}

void multiply(int key[SIZE][SIZE], int vector[SIZE], int result[SIZE]) {

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

result[i] = 0;

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

result[i] += key[i][j] \* vector[j];

}

result[i] %= 26;

}

}

int main() {

char message[100];

int key[SIZE][SIZE];

int choice;

printf("Enter 2x2 key matrix (only numbers):\n");

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

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

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

printf("1. Encrypt\n2. Decrypt\nEnter your choice: ");

scanf("%d", &choice);

printf("Enter message (2 letters, only A-Z): ");

scanf("%s", message);

int vector[SIZE], result[SIZE];

vector[0] = charToNum(message[0]);

vector[1] = charToNum(message[1]);

if (choice == 1) {

multiply(key, vector, result);

printf("Encrypted Text: %c%c\n", numToChar(result[0]), numToChar(result[1]));

} else if (choice == 2) {

int inv[SIZE][SIZE];

if (!inverseKey(key, inv)) {

printf("Key matrix is not invertible modulo 26.\n");

return 1;

}

multiply(inv, vector, result);

printf("Decrypted Text: %c%c\n", numToChar(result[0]), numToChar(result[1]));

} else {

printf("Invalid choice.\n");

}

return 0;

}

