**LAB\_31/01/2025**

**1. Write a C program for Caesar cipher involves replacing each letter of the alphabet with**

**the letter standing k places further down the alphabet, for k in the range 1 through 25.**

**CODE:**#include <stdio.h>

void caesarcipher(char\* text, int key)

{

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

{

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

{

text[i] = (text[i] - 'a' + key) % 26 + 'a';

} else if (text[i] >= 'A' && text[i] <= 'Z')

{

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

}

}

}

int main()

{

char text[100];

int key;

printf("Enter the text: ");

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

printf("Enter the key value: ");

scanf("%d", &key);

caesarcipher(text, key);

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

return 0;

}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**2. Write a C program for monoalphabetic substitution cipher maps a plaintext alphabet to a**

**ciphertext alphabet, so that each letter of the plaintext alphabet maps to a single unique**

**letter of the ciphertext alphabet.**

**CODE:**

#include <stdio.h>

#include <string.h>

#define ALPHABET "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

#define CIPHER "QWERTYUIOPLKJHGFDSAZXCVBNM"

void encryptMonoalphabetic(char text[]) {

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

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

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

else if (text[i] >= 'a' && text[i] <= 'z')

text[i] = CIPHER[text[i] - 'a'] + 32;

}

}

int main() {

char text[100];

printf("Enter text: ");

gets(text);

encryptMonoalphabetic(text);

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

return 0;

}

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**3.Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters**

**constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.**

**CODE**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 5

char keyMatrix[SIZE][SIZE];

void prepareKeyMatrix(char key[]) {

int used[26] = {0}, row = 0, col = 0;

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

char ch = toupper(key[i]) == 'J' ? 'I' : toupper(key[i]);

if (!used[ch - 'A']) keyMatrix[row][col++] = ch, used[ch - 'A'] = 1, col %= SIZE, row += !col;

}

for (char ch = 'A'; ch <= 'Z'; ch++) if (ch != 'J' && !used[ch - 'A'])

keyMatrix[row][col++] = ch, used[ch - 'A'] = 1, col %= SIZE, row += !col;

}

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

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

if (keyMatrix[i][j] == ch) \*row = i, \*col = j;

}

void prepareText(char text[], char prepared[]) {

int len = 0;

for (int i = 0; text[i]; i++) if (text[i] != ' ') prepared[len++] = toupper(text[i]);

prepared[len] = '\0';

char temp[100]; int j = 0;

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

temp[j++] = prepared[i];

temp[j++] = (i + 1 < len && prepared[i] != prepared[i + 1]) ? prepared[i + 1] : 'X';

}

if (j % 2) temp[j++] = 'X';

temp[j] = '\0';

strcpy(prepared, temp);

}

void encryptText(char text[]) {

char encrypted[100]; int j = 0;

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

int r1, c1, r2, c2;

findPosition(text[i], &r1, &c1);

findPosition(text[i + 1], &r2, &c2);

if (r1 == r2) encrypted[j++] = keyMatrix[r1][(c1 + 1) % SIZE], encrypted[j++] = keyMatrix[r2][(c2 + 1) % SIZE];

else if (c1 == c2) encrypted[j++] = keyMatrix[(r1 + 1) % SIZE][c1], encrypted[j++] = keyMatrix[(r2 + 1) % SIZE][c2];

else encrypted[j++] = keyMatrix[r1][c2], encrypted[j++] = keyMatrix[r2][c1];

}

encrypted[j] = '\0';

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

}

int main() {

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

printf("Enter keyword: "), gets(key), prepareKeyMatrix(key);

printf("Enter plaintext: "), gets(text), prepareText(text, prepared);

encryptText(prepared);

return 0;

}

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**4. As you know, the most frequently occurring letter in English is e. Therefore, the first or**

**second (or perhaps third?) most common character in the message is likely to stand for e.**

**Also, e is often seen in pairs (e.g., meet, fleet, speed, seen, been, agree, etc.). Try to find a**

**character in the ciphertext that decodes to e.**

**2. The most common word in English is “the.” Use this fact to guess the characters**

**that stand for t and h. 3. Decipher the rest of the message by deducing additional**

**words.**

**CODE:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_LEN 100

void frequencyAnalysis(char \*ciphertext) {

int freq[26] = {0};

int i, maxIndex = 0, secondMaxIndex = 0, thirdMaxIndex = 0;

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

if (isalpha(ciphertext[i])) {

freq[ciphertext[i] - 'A']++;

}

}

for (i = 1; i < 26; i++) {

if (freq[i] > freq[maxIndex]) {

thirdMaxIndex = secondMaxIndex;

secondMaxIndex = maxIndex;

maxIndex = i;

} else if (freq[i] > freq[secondMaxIndex]) {

thirdMaxIndex = secondMaxIndex;

secondMaxIndex = i;

} else if (freq[i] > freq[thirdMaxIndex]) {

thirdMaxIndex = i;

}

}

printf("Most frequent letters: %c, %c, %c (likely mapping to E, T, H)\n", maxIndex + 'A', secondMaxIndex + 'A', thirdMaxIndex + 'A');

}

void decryptUsingSubstitution(char \*ciphertext, char \*decryptedText, char map[26]) {

int i;

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

if (isalpha(ciphertext[i])) {

decryptedText[i] = map[ciphertext[i] - 'A'];

} else {

decryptedText[i] = ciphertext[i];

}

}

decryptedText[i] = '\0';

}

int main() {

char ciphertext[MAX\_LEN] = "GUVF VF N FRPERG ZRFFNTR"; // Example cipher text

char decryptedText[MAX\_LEN];

char substitutionMap[26] = {'T', 'H', 'E', 'A', 'O', 'N', 'I', 'S', 'R', 'D', 'L', 'C', 'U', 'M', 'W', 'F', 'G', 'Y', 'P', 'B', 'V', 'K', 'J', 'X', 'Q', 'Z'};

frequencyAnalysis(ciphertext);

decryptUsingSubstitution(ciphertext, decryptedText, substitutionMap);

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

return 0;

}

**OUTPUT** A screenshot of a computer

AI-generated content may be incorrect.

**5. Write a C program for monoalphabetic cipher is that both sender and receiver must commit the permuted cipher sequence to memory. A common technique for avoiding this is to use a keyword from which the cipher sequence can be generated. For example, using the keyword CIPHER, write out the keyword followed by unused letters in normal order and match this against the plaintext letters:**

**plain: 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**

**cipher: C I P H E R A B D F G J K L M N O Q S T U V W X Y Z**

**CODE**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define ALPHABET\_SIZE 26

void generateCipherAlphabet(char \*keyword, char \*cipherAlphabet) {

int used[ALPHABET\_SIZE] = {0};

int i, j = 0, len = strlen(keyword);

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

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

cipherAlphabet[j++] = keyword[i];

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

}

}

for (i = 0; i < ALPHABET\_SIZE; i++) {

if (!used[i]) {

cipherAlphabet[j++] = 'A' + i;

}

}

cipherAlphabet[j] = '\0';

}

void monoalphabeticEncrypt(char \*plaintext, char \*cipherAlphabet, char \*ciphertext) {

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

if (isalpha(plaintext[i])) {

ciphertext[i] = cipherAlphabet[toupper(plaintext[i]) - 'A'];

} else {

ciphertext[i] = plaintext[i];

}

}

ciphertext[strlen(plaintext)] = '\0';

}

void monoalphabeticDecrypt(char \*ciphertext, char \*cipherAlphabet, char \*decryptedText) {

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

if (isalpha(ciphertext[i])) {

for (int j = 0; j < ALPHABET\_SIZE; j++) {

if (cipherAlphabet[j] == ciphertext[i]) {

decryptedText[i] = 'A' + j;

break;

}

}

} else {

decryptedText[i] = ciphertext[i];

}

}

decryptedText[strlen(ciphertext)] = '\0';

}

int main() {

char keyword[] = "CIPHER";

char cipherAlphabet[ALPHABET\_SIZE + 1];

char plaintext[] = "MEET ME AT THE USUAL PLACE AT TEN RATHER THAN EIGHT OCLOCK";

char ciphertext[100], decryptedText[100];

generateCipherAlphabet(keyword, cipherAlphabet);

monoalphabeticEncrypt(plaintext, cipherAlphabet, ciphertext);

monoalphabeticDecrypt(ciphertext, cipherAlphabet, decryptedText);

printf("Cipher Alphabet: %s\n", cipherAlphabet);

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

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

return 0;

}

**OUTPUT**

**A screenshot of a computer

AI-generated content may be incorrect.**

**6. Write a C program for Playfair matrix:**

A close-up of a letter

AI-generated content may be incorrect.

**Encrypt this message: Must see you over Cadogan West. Coming at once.**

**CODE:**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 5

void generate\_playfair\_matrix(char keyword[], char matrix[SIZE][SIZE]) {

char alphabet[] = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

char key\_processed[26] = "";

int used[26] = {0};

int k = 0, index = 0;

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

char ch = toupper(keyword[i]);

if (ch == 'J') ch = 'I';

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

key\_processed[k++] = ch;

used[ch - 'A'] = 1;

}

}

for (int i = 0; i < 25; i++) {

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

key\_processed[k++] = alphabet[i];

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

}

}

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

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

matrix[i][j] = key\_processed[index++];

}

}

}

void print\_matrix(char matrix[SIZE][SIZE]) {

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

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

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

}

printf("\n");

}

}

int main() {

char keyword[] = "CIPHER";

char playfair\_matrix[SIZE][SIZE];

generate\_playfair\_matrix(keyword, playfair\_matrix);

printf("Playfair Cipher Matrix:\n");

print\_matrix(playfair\_matrix);

return 0;

}

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**7 Write a C program to Encrypt the message “meet me at the usual place at ten rather than**

**eight oclock” using the Hill cipher with the key.**

**( 9 4 )**

**(5 7 )**

**a. Show your calculations and the result. b. Show the calculations for the corresponding**

**decryption of the ciphertext to recover the original plaintext**

**CODE**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define SIZE 2

#define MOD 26

int key[SIZE][SIZE] = {{9, 4}, {5, 7}};

int inverseKey[SIZE][SIZE] = {{7, -4}, {-5, 9}};

void prepareText(char \*text) {

int i, len = strlen(text), j = 0;

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

if (isalpha(text[i])) {

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

}

}

text[j] = '\0';

if (j % SIZE != 0) {

text[j++] = 'X';

text[j] = '\0';

}

}

void multiplyMatrix(int key[SIZE][SIZE], int textVector[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] \* textVector[j];

}

result[i] %= MOD;

if (result[i] < 0) result[i] += MOD;

}

}

void hillCipherEncrypt(char \*plaintext, char \*ciphertext) {

int len = strlen(plaintext), vector[SIZE], result[SIZE];

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

vector[0] = plaintext[i] - 'A';

vector[1] = plaintext[i + 1] - 'A';

multiplyMatrix(key, vector, result);

ciphertext[i] = result[0] + 'A';

ciphertext[i + 1] = result[1] + 'A';

}

ciphertext[len] = '\0';

}

void hillCipherDecrypt(char \*ciphertext, char \*decryptedText) {

int len = strlen(ciphertext), vector[SIZE], result[SIZE];

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

vector[0] = ciphertext[i] - 'A';

vector[1] = ciphertext[i + 1] - 'A';

multiplyMatrix(inverseKey, vector, result);

decryptedText[i] = result[0] + 'A';

decryptedText[i + 1] = result[1] + 'A';

}

decryptedText[len] = '\0';

}

int main() {

char plaintext[] = "MEET ME AT THE USUAL PLACE AT TEN RATHER THAN EIGHT OCLOCK";

char ciphertext[100], decryptedText[100];

prepareText(plaintext);

hillCipherEncrypt(plaintext, ciphertext);

hillCipherDecrypt(ciphertext, decryptedText);

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

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

return 0;

}

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.