```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 10
#define ALPHABET_SIZE 64

int pgcd(int a, int b) {
    while (b != 0) {
        int r = a % b;
        a = b;
        b = r;
    }
    return a;
}

int Inversedet(int d) {
    int r = d % ALPHABET_SIZE;
    if (r < 0) r += ALPHABET_SIZE;
    return r;
}

void getCofactor(int K[MAX][MAX], int temp[MAX][MAX], int p, int q, int
n) {
    int i = 0, j = 0;
    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
            if (row != p && col != q) {
                temp[i][j++] = K[row][col];
                if (j == n - 1) {
                    j = 0;
                    i++;
                }
            }
        }
    }
}

int determinant(int mat[MAX][MAX], int n) {
    if (n == 1)
        return mat[0][0];
    int temp[MAX][MAX], sign = 1, det = 0;
    for (int f = 0; f < n; f++) {
        getCofactor(mat, temp, 0, f, n);
        det += sign * mat[0][f] * determinant(temp, n - 1);
        sign = -sign;
    }
    return det;
}

void comatrice(int K[MAX][MAX], int C[MAX][MAX], int n) {
    int temp[MAX][MAX];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            getCofactor(K, temp, i, j, n);
            C[i][j] = ((i + j) % 2 == 0 ? 1 : -1) * determinant(temp, n -
1);
        }
    }
```

```
}

void transpose(int mat[MAX][MAX], int result[MAX][MAX], int n) {
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            result[j][i] = mat[i][j];
}

void InverseMat(int K[MAX][MAX], int result[MAX][MAX], int n) {
    int det = determinant(K, n);
    if (pgcd(det, ALPHABET_SIZE) != 1) {
        printf("La matrice n'est pas inversible modulo %d.\n",
ALPHABET_SIZE);
        return;
    }

    int detInverse = -1;
    for (int i = 1; i < ALPHABET_SIZE; i++) {
        if ((det * i) % ALPHABET_SIZE == 1) {
            detInverse = i;
            break;
        }
    }

    if (detInverse == -1) {
        printf("Pas d'inverse modulaire trouvÃ© pour le
dÃ©terminant.\n");
        return;
    }

    int com[MAX][MAX], adj[MAX][MAX];
    comatrice(K, com, n);
    transpose(com, adj, n);

    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j) {
            result[i][j] = (adj[i][j] * detInverse) % ALPHABET_SIZE;
            if (result[i][j] < 0)
                result[i][j] += ALPHABET_SIZE;
        }
}

int charToIndex(char c) {
    if (c >= 'A' && c <= 'Z') return c - 'A';              // 0â€“25
    if (c >= 'a' && c <= 'z') return c - 'a' + 26;         // 26â€“51
    if (c >= '1' && c <= '9') return c - '1' + 52;         // 52â€–60
    if (c == '0') return 61;                               // 61
    if (c == ' ') return 62;                               // 62
    return 63;                                             // 'X' par
dÃ©faut
}

char indexToChar(int idx) {
    if (idx >= 0 && idx < 26) return 'A' + idx;
    if (idx >= 26 && idx < 52) return 'a' + (idx - 26);
    if (idx >= 52 && idx < 61) return '1' + (idx - 52);
    if (idx == 61) return '0';
    if (idx == 62) return ' ';
```

```
    return 'X';
}

void texteVersVecteurs(char* texte, int vecteurs[][MAX], int taille, int*
nbVecteurs) {
    int len = strlen(texte);
    *nbVecteurs = (len + taille - 1) / taille;

    int i, j = 0, k = 0;
    for (i = 0; i < *nbVecteurs * taille; ++i) {
        if (i < len)
            vecteurs[j][k++] = charToIndex(texte[i]);
        else
            vecteurs[j][k++] = charToIndex('X');
        if (k == taille) {
            j++;
            k = 0;
        }
    }
}

void vecteursVersTexte(int vecteurs[][MAX], int nbVecteurs, int taille,
char* resultat) {
    int index = 0;
    for (int i = 0; i < nbVecteurs; ++i)
        for (int j = 0; j < taille; ++j)
            resultat[index++] = indexToChar(vecteurs[i][j]);
    resultat[index] = '\0';
}

void chiffrer(int K[MAX][MAX], int vecteurs[][MAX], int nbVecteurs, int
taille, int resultat[][MAX]) {
    for (int v = 0; v < nbVecteurs; ++v) {
        for (int i = 0; i < taille; ++i) {
            resultat[v][i] = 0;
            for (int j = 0; j < taille; ++j)
                resultat[v][i] += vecteurs[v][j] * K[i][j];
            resultat[v][i] %= ALPHABET_SIZE;
        }
    }
}

void dechiffrer(int Kinv[MAX][MAX], int vecteurs[][MAX], int nbVecteurs,
int taille, int resultat[][MAX]) {
    for (int v = 0; v < nbVecteurs; ++v) {
        for (int i = 0; i < taille; ++i) {
            resultat[v][i] = 0;
            for (int j = 0; j < taille; ++j)
                resultat[v][i] += vecteurs[v][j] * Kinv[i][j];
            resultat[v][i] %= ALPHABET_SIZE;
        }
    }
}

int main() {

    int choix;
    char texte[1000];
```

```
int a, b;

do {
    printf("\n\n\t\tGroupe: Fadel && Madeleine");
    printf("\n\n\t\tMenu\n");
    printf("\t\t\t1. Chiffrer\n");
    printf("\t\t\t2. Dechiffrer\n");
    printf("\t\t\t3. Quitter\n");
    printf("\t\t\tVotre choix : ");
    scanf("%d", &choix);
    getchar();

    switch (choix) {
        case 1:
            {
                int n;
                printf("Entrer le nombre de ligne de (n x n): ");
                scanf("%d", &n);

                int K[MAX][MAX];
                printf("Entrer les element de la matrice:\n");
                for (int i = 0; i < n; ++i) {
                    for (int j = 0; j < n; ++j) {
                        printf("%d%d: ", i, j);
                        scanf("%d", &K[i][j]);
                    }
                }

                char message[MAX];
                printf("Entrez le message a crypter: ");
                scanf("%s", message);

                int vecteurs[100][MAX], cryptes[100][MAX];
                int nbVecteurs;

                texteVersVecteurs(message, vecteurs, n, &nbVecteurs);
                chiffrer(K, vecteurs, nbVecteurs, n, cryptes);

                char resultat[100];
                vecteursVersTexte(cryptes, nbVecteurs, n, resultat);

                printf("Message crypte :%s\n", resultat);
            }
            break;

        case 2:
            {
                int n;
                printf("Entrer le nombre de ligne de (n x n): ");
                scanf("%d", &n);

                int K[MAX][MAX];
                printf("Entrer les element de la matrice:\n");
                for (int i = 0; i < n; ++i) {
                    for (int j = 0; j < n; ++j) {
                        printf("%d%d: ", i, j);
                        scanf("%d", &K[i][j]);
                    }
```

```
                    }

                    char message[MAX];
                    printf("Entrez le message a dechiffrer: ");
                    scanf("%s", message);

                    int Kinv[MAX][MAX];
                    InverseMat(K, Kinv, n);

                    int vecteurs[100][MAX], dechiffres[100][MAX];
                    int nbVecteurs;

                    texteVersVecteurs(message, vecteurs, n, &nbVecteurs);
                    dechiffrer(Kinv, vecteurs, nbVecteurs, n,
dechiffres);

                    char resultat_dechiffre[100];
                    vecteursVersTexte(dechiffres, nbVecteurs, n,
resultat_dechiffre);

                    printf("Message dechiffre : %s\n",
resultat_dechiffre);
                }
                break;

            case 3:
                printf("Au revoir !\n");
                break;

            default:
                printf("Choix invalide.\n");
        }
    } while (choix != 3);

    return 0;
}
```