```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX 1000
#define MOD 37

// Convertissons un caractère en nombre (Z37)
int char_to_code(char c)
{
    if (c >= 'a' && c <= 'z')
    {
        return c - 'a';
    }
    else if (c >= 'A' && c <= 'Z')
    {
        return c - 'A';
    }
    else if (c >= '0' && c <= '9')
    {
        return c - '0' + 26;
    }
    else if (c == ' ')
    {
        return 36;
    }
    else if (c == 'X' || c == 'x') // Pour le remplissage
    {
        return 23; // Code de X
    }
    else
    {
        return 0;
    }
}

// Convertissons un nombre en caractère (Z37)
char code_to_char(int code)
{
    if (code >= 0 && code <= 25)
    {
        return 'A' + code;
    }
    else if (code >= 26 && code <= 35)
    {
        return '0' + (code - 26);
    }
    else if (code == 36)
    {
        return ' '; // Utiliser espace au lieu de '_'
    }
    else
    {
        return '?';
    }
}

// Calculons du PGCD de deux nombres
```

```
int pgcd(int a, int b)
{
    if (b == 0)
        return a;
    return pgcd(b, a % b);
}

// Calculons  l'inverse modulaire (a^-1 mod m)
int inverse_mod(int a, int m)
{
    int i;
    for (i = 1; i < m; i++)
    {
        if ((a * i) % m == 1)
            return i;
    }
    return -1; // Pas d'inverse
}

// Calculons le déterminant d'une matrice 2x2
int det2(int mat[5][5])
{
    return (mat[0][0] * mat[1][1] - mat[0][1] * mat[1][0]) % MOD;
}

// Calculons le déterminant d'une matrice 3x3
int det3(int mat[5][5])
{
    return (mat[0][0] * (mat[1][1] * mat[2][2] - mat[1][2] * mat[2][1]) -
            mat[0][1] * (mat[1][0] * mat[2][2] - mat[1][2] * mat[2][0]) +
            mat[0][2] * (mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0]))
% MOD;
}

// Calculons de la matrice inverse pour une matrice 2x2
void inverse_matrice2(int mat[5][5], int inv[5][5])
{
    int det = det2(mat);
    if (det < 0)
        det += MOD;

    int det_inv = inverse_mod(det, MOD);

    if (det_inv == -1)
    {
        printf("La matrice n'est pas inversible dans Z%d\n", MOD);
        exit(1);
    }

    inv[0][0] = (mat[1][1] * det_inv) % MOD;
    inv[0][1] = ((-mat[0][1] + MOD) * det_inv) % MOD;
    inv[1][0] = ((-mat[1][0] + MOD) * det_inv) % MOD;
    inv[1][1] = (mat[0][0] * det_inv) % MOD;
}

// Calculons  la matrice inverse pour une matrice 3x3
void inverse_matrice3(int mat[5][5], int inv[5][5])
{
```

```
    int det = det3(mat);
    if (det < 0)
        det += MOD;

    int det_inv = inverse_mod(det, MOD);

    if (det_inv == -1)
    {
        printf("La matrice n'est pas inversible dans Z%d\n", MOD);
        exit(1);
    }

    // Calculons les cofacteurs
    inv[0][0] = ((mat[1][1] * mat[2][2] - mat[1][2] * mat[2][1]) *
det_inv) % MOD;
    inv[0][1] = ((mat[0][2] * mat[2][1] - mat[0][1] * mat[2][2]) *
det_inv) % MOD;
    inv[0][2] = ((mat[0][1] * mat[1][2] - mat[0][2] * mat[1][1]) *
det_inv) % MOD;

    inv[1][0] = ((mat[1][2] * mat[2][0] - mat[1][0] * mat[2][2]) *
det_inv) % MOD;
    inv[1][1] = ((mat[0][0] * mat[2][2] - mat[0][2] * mat[2][0]) *
det_inv) % MOD;
    inv[1][2] = ((mat[0][2] * mat[1][0] - mat[0][0] * mat[1][2]) *
det_inv) % MOD;

    inv[2][0] = ((mat[1][0] * mat[2][1] - mat[1][1] * mat[2][0]) *
det_inv) % MOD;
    inv[2][1] = ((mat[0][1] * mat[2][0] - mat[0][0] * mat[2][1]) *
det_inv) % MOD;
    inv[2][2] = ((mat[0][0] * mat[1][1] - mat[0][1] * mat[1][0]) *
det_inv) % MOD;

    // S'assurons nous que tous les éléments sont positifs
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (inv[i][j] < 0)
                inv[i][j] += MOD;
        }
    }
}

int main()
{
    int choix, taille, continuer = 1;
    int cle[5][5], cle_inv[5][5];
    char texte[MAX];
    int vecteur[5];
    int resultat[5];
    int i, j, k, len;
    char reponse;

    // Initialisons le générateur de nombres aléatoires
    srand(time(NULL));
```

```
    while (continuer)
    {
        printf("\n/// Chiffrement et Dechiffrement De Hill Dans Z37
///\n");
        printf("1. Chiffrer\n");
        printf("2. Dechiffrer\n");
        printf("Entrez votre choix (1 ou 2): ");
        scanf("%d", &choix);

        printf("Entrez la taille de la matrice cle (2 ou 3) : ");
        scanf("%d", &taille);

        if (taille != 2 && taille != 3)
        {
            printf("Taille invalide. Elle doit etre 2 ou 3.\n");
            continue;
        }

        // Saisisons de la matrice clé
        printf("Entrez les elements de la matrice cle %dx%d (dans Z37)
:\n", taille, taille);
        for (i = 0; i < taille; i++)
        {
            for (j = 0; j < taille; j++)
            {
                printf("Cle[%d][%d] = ", i, j);
                scanf("%d", &cle[i][j]);
                cle[i][j] = cle[i][j] % MOD;
                if (cle[i][j] < 0) cle[i][j] += MOD;
            }
        }

        // Calculons la matrice inverse pour le déchiffrement
        if (choix == 2)
        {
            if (taille == 2)
                inverse_matrice2(cle, cle_inv);
            else
                inverse_matrice3(cle, cle_inv);
        }

        // Nettoyons du buffer
        getchar();

        // Saisie du texte
        if (choix == 1)
            printf("Entrez le texte a chiffrer : ");
        else
            printf("Entrez le texte a dechiffrer : ");

        fgets(texte, MAX, stdin);

        // Calculons de la longueur du texte
        len = 0;
        while (texte[len] != '\0' && texte[len] != '\n')
        {
            len++;
        }
```

```
        // Ajout de 'X' si le texte n'est pas multiple de la taille pour
le chiffrement
        if (choix == 1)
        {
            while (len % taille != 0)
            {
                texte[len] = 'X';
                len++;
            }
            texte[len] = '\0';
        }

        if (choix == 1)
            printf("Texte chiffre : ");
        else
            printf("Texte dechiffre : ");

        // Traitement par blocs
        for (i = 0; i < len; i += taille)
        {
            // Remplir le vecteur
            for (j = 0; j < taille; j++)
            {
                if (i + j < len)
                    vecteur[j] = char_to_code(texte[i + j]);
                else
                    vecteur[j] = 0;
            }

            // Multiplier par la matrice (clé ou clé inverse)
            for (j = 0; j < taille; j++)
            {
                resultat[j] = 0;
                for (k = 0; k < taille; k++)
                {
                    if (choix == 1)
                        resultat[j] += cle[j][k] * vecteur[k];
                    else
                        resultat[j] += cle_inv[j][k] * vecteur[k];
                }
                resultat[j] = resultat[j] % MOD;
                if (resultat[j] < 0) resultat[j] += MOD;
            }

            // Afficher les caractères
            for (j = 0; j < taille; j++)
            {
                if (i + j < len)
                {
                    if (choix == 2 && resultat[j] == 36)
                    {
                        // Pour le déchiffrement, on affiche un espace
                        printf(" ");
                    }
                    else
                    {
                        printf("%c", code_to_char(resultat[j]));
```

```
                    }
                }
            }
        }

        printf("\n\nVoulez-vous continuer? (o/n): ");
        scanf(" %c", &reponse);
        if (reponse != 'o' && reponse != 'O')
        {
            continuer = 0;
        }
    }

    return 0;
}
```