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

#define MAX 100
#define ALPHABET_SIZE 37

void lireMatrice(int matrice[MAX][MAX], int taille) {
    printf("Entrez les éléments de la matrice clé (%dx%d):\n", taille,
taille);
    for (int i = 0; i < taille; i++) {
        for (int j = 0; j < taille; j++) {
            scanf("%d", &matrice[i][j]);
        }
    }
}

void lireMessage(char *msg) {
    printf("Entrez le texte à traiter (lettres, chiffres, espace): ");
    getchar(); // Ignore le retour à la ligne précédent
    fgets(msg, MAX, stdin);
    msg[strcspn(msg, "\n")] = 0; // Retirer le retour à la ligne
}

void ajusterMessage(char *msg, int tailleBloc) {
    int longueur = strlen(msg);
    while (longueur % tailleBloc != 0) {
        msg[longueur++] = ' '; // Ajouter un espace pour le padding
    }
    msg[longueur] = '\0';
}

int caractereVersNuPo(char caractere) {
    if (caractere >= 'A' && caractere <= 'Z') return caractere - 'A';
    if (caractere >= '0' && caractere <= '9') return caractere - '0' +
26;
    if (caractere == ' ') return 36; // Espace
    return -1; // Caractère non valide
}

char nuPoVersCaractere(int NuPo) {
    if (NuPo >= 0 && NuPo < 26) return 'A' + NuPo;
    if (NuPo >= 26 && NuPo < 36) return '0' + (NuPo - 26);
    if (NuPo == 36) return ' ';
    return '?'; // Erreur pour les valeurs hors limites
}

void chiffrementHill(int clé[MAX][MAX], char *entrée, char *sortie, int
taille) {
    int longueur = strlen(entrée);
    for (int i = 0; i < longueur; i += taille) {
        for (int j = 0; j < taille; j++) {
            sortie[i + j] = 0;
            for (int k = 0; k < taille; k++) {
                int NuPo = caractereVersNuPo(entrée[i + k]);
                if (NuPo != -1) {
                    sortie[i + j] += clé[j][k] * NuPo;
                }
```

```
            }
            sortie[i + j] = sortie[i + j] % ALPHABET_SIZE; // Assurer que
la valeur reste dans les limites
        }
    }
    sortie[longueur] = '\0';
}

void inverserMatrice(int clé[MAX][MAX], int inverse[MAX][MAX], int
taille) {
    // Implémenter l'inversion de la matrice ici si nécessaire
}

void dechiffrementHill(int clé[MAX][MAX], char *entrée, char *sortie, int
taille) {
    int inverse[MAX][MAX];
    inverserMatrice(clé, inverse, taille);

    int longueur = strlen(entrée);
    for (int i = 0; i < longueur; i += taille) {
        for (int j = 0; j < taille; j++) {
            sortie[i + j] = 0;
            for (int k = 0; k < taille; k++) {
                int NuPo = entrée[i + k];
                sortie[i + j] += inverse[j][k] * NuPo;
            }
            sortie[i + j] = (sortie[i + j] + ALPHABET_SIZE) %
ALPHABET_SIZE; // Assurer un résultat positif
        }
    }
    sortie[longueur] = '\0';
}

int main() {
    int clé[MAX][MAX], taille;
    char message[MAX], résultat[MAX];
    int choix;

    while (1) {
        // Saisir la taille de la matrice
        printf("Entrez la taille de la matrice (n x n): ");
        scanf("%d", &taille);

        // Lire la matrice clé
        lireMatrice(clé, taille);

        // Choix de l'utilisateur
        printf("Choisissez une option:\n1. Chiffrer\n2. Déchiffrer\n3.
Quitter\nVotre choix: ");
        scanf("%d", &choix);

        if (choix == 3) {
            printf("Au revoir !\n");
            break; // Quitter la boucle et le programme
        }

        // Lire le message à traiter
        lireMessage(message);
```

```
        ajusterMessage(message, taille);

        if (choix == 1) {
            // Chiffrement
            chiffrementHill(clé, message, résultat, taille);
            printf("Message chiffré : ");
        } else if (choix == 2) {
            // Déchiffrement
            dechiffrementHill(clé, message, résultat, taille);
            printf("Message déchiffré : ");
        } else {
            printf("Choix invalide.\n");
            continue; // Recommencer la boucle
        }

        for (int i = 0; i < strlen(message); i++) {
            printf("%c", nuPoVersCaractere(résultat[i]));
        }
        printf("\n");
    }

    return 0;
}
```