```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

// For the sake of simplicity ,
// * this implementation does not conserve the original case of the message ,
// * key is assumed to contain alphatic characters only ,
// * malloc is assumed to never fail .

void encrypt_1( char *message , const char *key , char (*vtab)[26]) {
   int iKey = 0;
   int iMessage = 0;
   while ( message[iMessage] != '\0') {
      const char cKey = tolower(key[iKey]);
      const char cMessage = tolower(message[iMessage]);

      if(cMessage >= 'a' && cMessage <= 'z')
      {
         message[iMessage] = vtab[cMessage -'a'][cKey - 'a'];
         ++ iKey ;
         if ( key[iKey] == '\0')
         {
            iKey = 0;
         }
      }
      ++ iMessage ;
   }
}

void decrypt_1 ( char *encrypted , const char *key , char (* vtab)[26]) {
   int iKey = 0;
   int iEncrypted = 0;
   while ( encrypted[iEncrypted] != '\0') {
      const char cKey = tolower( key[iKey]);
      const char cEncrypted = tolower( encrypted[iEncrypted]);
      if ( cEncrypted >= 'a' && cEncrypted <= 'z') {
         for (int i = 0; i < 26; ++i) {
            //Parcours colonne cKey a la recherche de cEncrypted
            if ( vtab[i][cKey -'a'] == cEncrypted )
            {
               encrypted[iEncrypted] = (char)('a' + i); // cEncrypted trouver en ligne i
            }
         }
         ++ iKey ;
         if (key[iKey] == '\0') {
            iKey = 0;
         }
      }
      ++ iEncrypted ;
   }
}
```

```
void encrypt_2 ( char * message , const char * key , char ** vtab ) {
   int iKey = 0;
   int iMessage = 0;
   while ( message[iMessage] != '\0') {
      const char cKey = tolower(key[iKey]);
      const char cMessage = tolower(message[iMessage]);

      if(cMessage >= 'a' && cMessage <= 'z')
      {
         message[iMessage] = vtab[cMessage -'a'][cKey - 'a'];
         ++iKey ;
         if ( key[iKey] == '\0')
         {
            iKey = 0;
         }
      }
      ++ iMessage ;
   }
}

void decrypt_2 ( char * encrypted , const char * key , char ** vtab ) {
  int iKey = 0;
   int iEncrypted = 0;
   while ( encrypted[iEncrypted] != '\0') {
      const char cKey = tolower( key[iKey]);
      const char cEncrypted = tolower( encrypted[iEncrypted]);
      if ( cEncrypted >= 'a' && cEncrypted <= 'z') {
         for (int i = 0; i < 26; ++i) {
            if ( vtab[i][cKey -'a'] == cEncrypted ) {
               encrypted[iEncrypted] = (char)('a' + i);
            }
         }
         ++ iKey ;
         if (key[iKey] == '\0') {
            iKey = 0;
         }
      }
      ++ iEncrypted ;
   }
}
```

```
void encrypt_3 ( char * message , const char * key ) {
    int iKey = 0;
    int iMessage = 0;
    while ( message [ iMessage ] != '\0') {
        const char cKey = tolower(key[iKey]);
        const char cMessage = tolower( message[iMessage]);
        if ( cMessage >= 'a' && cMessage <= 'z') {
            message[iMessage] = (char)( 'a' + (cMessage - 'a' + cKey - 'a')%26);
            ++ iKey ;
            if ( key[iKey] == '\0') {
                iKey = 0;
            }
        }
        ++ iMessage ;
    }
}

void decrypt_3 ( char *encrypted , const char *key ) {
    int iKey = 0;
    int iEncrypted = 0;
    while ( encrypted [ iEncrypted ] != '\0') {
        const char cKey = tolower(key[iKey]);
        const char cEncrypted = tolower( encrypted[iEncrypted]);
        if ( cEncrypted >= 'a' && cEncrypted <= 'z') {
            // num colonne = cEncrypted - 'a' //  'a' - ckey = num colonne
            encrypted[iEncrypted] = (char)( 'a'+ (26 + cEncrypted -'a' - cKey + 'a')%26);
            ++ iKey ;
            if ( key[iKey] == '\0') {
                iKey = 0;
            }
        }
        ++ iEncrypted ;
    }
}
```

```
int main () {
    char message[] = "L'escargot se promene avec sa maison";
    const char *key = "perdu";

    // Q. 10
    char vigenereTable1[26][26];
    for (int i = 0; i < 26; ++i) {
        char letter = (char)('a' + i);
        for (int j = 0; j < 26; ++j) {
            vigenereTable1[i][j] = letter ;
            ++letter ;
            if ( letter > 'z') {
                letter = 'a';
            }
        }
    }
    encrypt_1( message , key , vigenereTable1);
    printf (" encrypt1 : %s\n", message);
    decrypt_1( message , key , vigenereTable1);
    printf(" decrypt1 : %s\n", message);

    // Q. 11
    char **vigenereTable2 = malloc (26*sizeof(char *));

    for (int i = 0; i < 26; ++i) {
        vigenereTable2[i] = malloc(26);

        char letter = (char)('a' + i);
        for (int j = 0; j < 26; ++j) {
            vigenereTable2[i][j] = letter ;
            ++letter ;
            if ( letter > 'z') {
                letter = 'a';
            }
        }
    }

    printf (" Message : %s\n", message );
    encrypt_2 ( message , key , vigenereTable2 );
    printf (" encrypt2 : %s \n", message );
    decrypt_2 ( message , key , vigenereTable2 );
    printf (" decrypt2 : %s\n", message );

    for (int i = 0; i < 26; ++i) {
        free(vigenereTable2[i]);
    }
    free(vigenereTable2);

    // Q. 12
    encrypt_3 ( message , key );
    printf (" encrypt3 : %s\n", message );
    decrypt_3 ( message , key );
    printf (" decrypt3 : %s\n", message );
    return 0;
}
```