1. **Données d’entrée et de sortie**

|  |  |
| --- | --- |
| Entrée | Sortie |
| message | cesar |
| clef |  |
|  |  |

1. **Interface**

Votre message chiffre est:

image

image

Introduire le message:

Clef:

CHIFFREMENT CESAR

1. **Algorithme**

Lis message, clef

Let alphabet 🡨 [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]

Let alphabet\_majuscule 🡨 [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]

1. Function eleve\_espace

Let indice🡨0

Let longueur 🡨 0

Let message\_sans\_espace = [ ]

Tant que message [indice] ! = undefined

SI message [indice] != " " {

message\_sans\_espace 🡨 message\_sans\_espace + message [indice]

longueur++ }

indice++

return message\_sans\_espace

1. Function chiffrement\_cesar

Let indice 🡨 0

Let cesar 🡨 " "

Let position 🡨 0

Tant que position < longueur

SI alphabet [indice] == message\_sans\_espace [position]

SI indice+clef >= maximo

let sommeindice 🡨 (indice + clef)-maximo

cesar 🡨 cesar + alphabet [sommeindice]

SI NON

cesar 🡨 cesar + alphabet [indice + clef]

position++

indice = 0;

Cesar 🡨 cesar + alphabet [indice + clef]

Position ++

Índice 🡨 0

SI NON

SI alphabetmajuscule [indice] == message\_sans\_espace [position]

Cesar 🡨 cesar + alphabet\_majuscule [indice + clef]

Position ++

Índice 🡨 0

Índice ++

1. Function dechiffrement\_cesar

Let indice 🡨 0

Let cesar 🡨 " "

Let position 🡨 0

Tant que position < longueur

SI alphabet [indice] == message\_sans\_espace [position]

SI indice - clef < 0

let sommeindice 🡨 (indice - clef) + maximo

cesar 🡨 cesar + alphabet [sommeindice]

SI NON

Cesar 🡨 cesar + alphabet [indice - clef]

Position ++

Índice 🡨 0

SI NON

SI alphabetmajuscule [indice] == message\_sans\_espace [position]

Cesar 🡨 cesar + alphabet\_majuscule [indice - clef]

Position ++

Índice 🡨 0

Índice ++