DS N°2: Corrigé

Ex 1 : (2,75p)

1) Il s'agit d'un traitement récurrent d'ordre deux, car pour calculer un terme de la suite on utilise deux termes précédents. (Ui= -Ui-1 + 2*Ui-2)

2) Fonction suite (n:entier): entier

```
Début
    U0← 0
    U1← 3
    pour i de 2 à (n-1) faire:
        U← -U1 + 2*U0
        U0← U1
        U1← U
    Fin pour
    Retourner U
Fin
```

T.D.O

O	T/N
U0	Entier
U1	Entier
U	
i	

Ex 2 : (2,75p)

1) si x=1 --> arctan(x)= Pi/4 --> Remplacer x par 1
Pi/4= 1-1/3+1/5-1/7+……

2) Fonction James (Epsilon : réel) : réel

```
Début
    S1←1
    i←1
    signe←-1
    répéter:
        S2←S1
        i←i+2
        S1← S2+signe*1/i
    Jusqu'a (ABS(4*S1-4*S2)<Epsilon)
     Retourner 4*S1
Fin
```

T.D.O

O	T/N
i	Entier
signe	Entier
S1	Réel
S2	Réel

Ex 3 : (5,5p)

Procédure Remplir (@F : Fich, @e : enreg)

```
Début
    Ouvrir ("Niven.dat", F, "wb") -
    Pour n de 100 à 1000 faire :
        Si niven(n) ET
    (EstNum(conv_dec_hex(n))= Faux)  Alors
        e. NV ← n
        e. NV_Hex ← conv_dec_hex(n)
    FinSi
    Ecrire (F,e)
    Fin
  Fermer (F)
Fin
```

T.D.N.T

Type
Enreg= Enregistrement
NV : entier
NV_Hex: chaine
Fin
Fich= fichier de Enreg

T.D. O

O	T/N
Conv_dec_hex	Fonction
n	Entier
Niven	Fonction

Algorithmes des modules utilisées:

Fonction Niven (n:entier): Booléen
Début
Retourner (n Mod Somme(n)=0)
Fin

T.D.O

O	T/N
Somme	Fonction

Fonction Somme (n:entier): entier
Début
Si n=0 Alors Retourner 0
Sinon Retourner (n Mod 10) + Somme (n div 10)
Fin Si
Fin

Solution étirative:
S←0
Répéter
S← S+ n Mod 10
N← N Div 10
Jusqu'a (N=0)
Retourner S

Ou encore :

Ch← convch(n)
Pour i de 0 à long(ch)-1 faire
S← s + val(ch[i])
Fin pour
Retourner S

T.D.O

O	T/N
S	entier

Fonction Conv_Dec_Hex(n :entier) : chaine
Début
CH← '0123456789ABCDEF'
Hex←''
Tantque (N ≠ 0) faire
Hex← CH[n Mod 16] + Hex
N← n Div 16
Fin
Retourner Hex
Fin

T.D.O

O	T/N
Ch, Hex	chaine

Ex 4 : (9 p)

Algorithme du Programme Principal:
Algorithme Cryptage
Début
Saisir (key, F1)
Remplir (key,F1,F2)
Afficher (F2)
Fin

Algorithmes des modules envisagés:
Procédure Afficher (@F2 :Texte)
Début
Ouvrir ("F_res.txt", F2,"r")
Tantque Non(Fin_fichier(F2)) faire
lire_ligne(F2,ch)
Ecrire(ch)
Fin
Fermer(F2)
Fin

T.D.O.G

O	T/N
Key	chaine
F1	Texte
F2	Texte
Saisir	Procédure
Remplir	Procédure
Afficher	Procédure

T.D.O

O	T/N
ch	chaine

Procédure Saisir (@key: chaine, @F1: Texte)

```
Début
        Ouvrir ("F_init.txt",F1,"r")
        Lire_ligne (F1,ch)
        K← ""
        Pour i de 0 à long(ch)-1 faire:
                K← K+ CHR(Aléa(65,91))
        Fin pour
        Key← Gen_Bin (k)
        Fermer (F1)
Fin
```

O	T/N
i ch,K Gen_Bin	entier chaine fonction

Procédure Remplir (key : chaine, @F1, F2 : Texte) **T.D.O**

```
Début
        Ouvrir('F_init.txt',F1,'r')
        Ouvrir('F_res.txt',F2,'w')
        Tantque Non(Fin_Fichier(F1)) faire
                Lire_ligne(F1,ligne)
                CH← Gen_Bin(ligne)
                CH_C← crypter(key,CH)
                Ecrire_nl(F2,CH_C)
        Fintq
        Fermer (F1)
        Fermer(F2)
Fin
```

O	T/N
ligne CH CH_C Gen_Bin Crypter	chaine chaine (Bin) chaine (crypter) Fonction Fonction

Fonction Crypter (CH1,CH2 : chaine) :chaine

```
Début
        CH← Ou_Exclusif(CH1,CH2)
        X← Aléa(65,91)
        CH_C← codage(X,CH)
        Retourner CH_C
Fin
```

O	T/N
CH C CH_C Ou_Exclusif Codage	chaine entier chaine Fonction Fonction

Fonction Ou_exclusif (CH1, CH2 : chaine)chaine :

```
Début
        CH← ''
        Pour i de 0 à long(CH1)-1 faire
                Si CH1[i] ≠ CH2[i] alors
                        CH← CH+'1'
                Sinon
                        CH← CH+'0'
                FinSi
        Finpour
        Retourner CH
Fin
```

O	T/N
Bin i	Chaine Entier

Fonction Codage (x : entier , CH :chaine) :chaine

```
Début
        CH_C←''
        Tantque (CH ≠'') faire :
                D← conv_Bin_Dec(sous_chaine(CH,0,8)
                CH_C←CH_C+CHR(D+x)
                Effacer(CH,0,8)
        Fin tant que   Retourner CH_C
Fin
```

O	T/N
CH_C D Conv_Bin_Dec	chaine Entier Fonction

Fonction Conv_Dec_Bin (N :entier) :chaine

Début

Bin← '00000000'

i←7

Tantque (N≠0) faire :

Si (N Mod 2 ≠ 0) Alors

Bin[i]← '1'

FinSi

i← i-1

N← N Div 2

FinTq

Retourner Bin

Fin

O	T/N
Bin i	Chaine Entier

➔ On accepte toute autre solution correcte

➔ Il faut vérifier que la chaine Bin est de 8 bits

Autrement :

Fonction Conv_Dec_Bin (N :entier) :chaine

Début

Bin ← ''

Répéter

Si (N Mod 2 =0) Alors

Bin← '0' + Bin

Sinon

Bin←'1'+Bin

FinSi

N← N Div 2

Jusqu'à (N=0)

Tantque (long(Bin)<8) faire :

Bin← '0' + Bin

Fin tantque

Retourner Bin

Fin

O	T/N
Bin i	Chaine Entier

Fonction Conv_Bin_Dec (ch : chaine) : Entier

Début

D← 0, p ← 1

Pour i de long(ch) -1 à 0 faire (pas = -1)

D ← D + p*Val(ch[i])

P ← p*2

Fin

Retourner D

Fin

O	T/N
D, p i	Entier Entier