Algo :

- bac 2019 :

Exo 1 :

```
Fonction what (a, b : réel) : réel
Debut
  si (a - b) >= 0 alors
    retourner a
  Sinon
    retourner what (b, a)
  Fin Si
Fin
```

① → Réel

② → (a - b) ≥ 0

③ → 12

④ → C

Exo 2 :

$f(x) = \frac{1}{x}$, $d \geq]0; +\infty[$: / $\int_1^a f(x)\,dx$:

1) Fonction surface (a : réel, n : entier) : réel // Rectangle a gauche :

```
Debut
  S ← 0
  x ← 1
  h ← (a - 1)/n
  pour i de 1 a n faire
    S ← S + 1/x
    x ← x + h
  fin pour
  retourner S x h
Fin
```

TDO		
S	réel	surface $\int_1^a f(x)\,dx$
x	réel	Val d'abscisse
h	réel	Longeur
i	entier	compteur

2) fonction Cdcul (a : réel, n : entier) : réel.

```
Debut
  eps ← 0,0001
  e ← a
  tantque abs(surface(a,n) - 1) > eps faire
      e ← e + eps
  Fin tantque
  retourner e
Fin
```

to check !

Var	type
e	réel
eps	réel
surface	fonction

Exo 3

```
Fonction pgcd(a, b : entier) : entier
  Debut
    n ← a mod b
    si n = 0 alors
      retourner b
    fin si
    retourner pgcd(b, n)
  Fin
```

Var	type
n	entier

```
procedure Inv():
  Debut
    ouvrir(source, "Fraction.dat", "rb")
    ouvrir(Res, "Irreduct.dat", "wb")
    tantque non fin_fichier(source)
      lire(source, objet)
      Si pgcd(objet.Num, objet.Denom) = 1 alors
        ecrire(Res, objet)
      Fin si
```

TDOL

Var	type
source	fich
Res	fich
objet	frac
pgcd	fonction

fermer (Res)

fermer (source)

Fin

TDONT

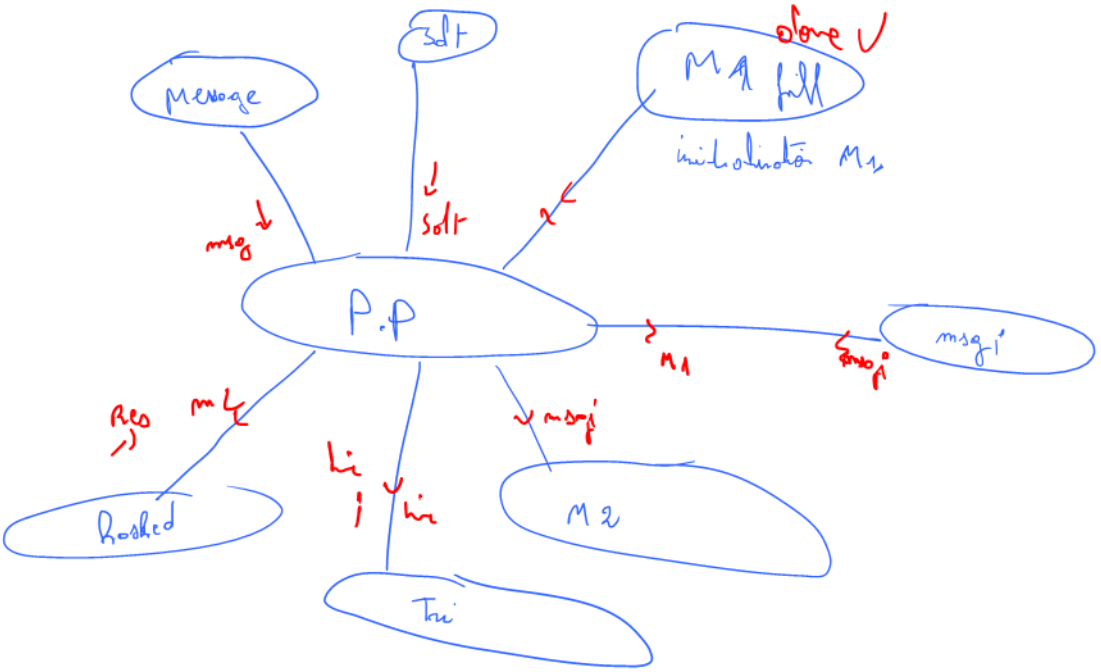
frac = enregistrement

Num : entier

Denom : entier

fich = fichier de type frac

problème



Algorithme Hash

Debut
Message ← msg( )
salt ← key( )
initialisation(M1)
Message2 ← msgi(M1, Message)
FillM(M2, Message2, salt)
Sort(M2)
Res ← hashed(M2)
Fin

TDOG

objet	type
Message, Message2, Salt, Res	chaine de char
M1	mot 1
M2	mot 2
msg, key, msgi, hashed	Fonction
initialisation, FillM, sort,	procedure

TDONTG

M1 : tableau 6x6 de character
M2 : tableau 6x7 de character

---

module Saisie de message :

Fonction msg( ) : chaine
Debut
Repeter
good ← vrai
Lire(ms)
pour i de 0 à Long(ms) - 1 faire
si ord(ms[i]) ∉ [97, 122] alors
good ← faux
fin si
fin pour
good ← Long[ms] ≤ 18
jusqu'a good

Retourner ms

Fin

TDOL

good	booleen
ms	chaine

Fonction soft( ) : chaine de charactère :

Debut

good ← vrai

Repeter

lire (s)

pour i de 0 à Long(s) - 1 faire

si ord [s[i]) ∉ [ord ("A"); ord ("B")] alors

good ← faux

fin si

Fin pour

good ← Long(s) = 6

jusqu'à good

retourner s

Fin

TDOL	
objet	type
s	chaine de charc
good	booleen

Fonction msg (M : mot1) : chaine de charactère

Debut

0 1 2 3 4 5

A B C D E F

1 2 3 4 5 6

ord A = 65

ord Z = 90

65 + 0 = A

65 + 1 = B

65 + 2 = C

65 + 3 = D

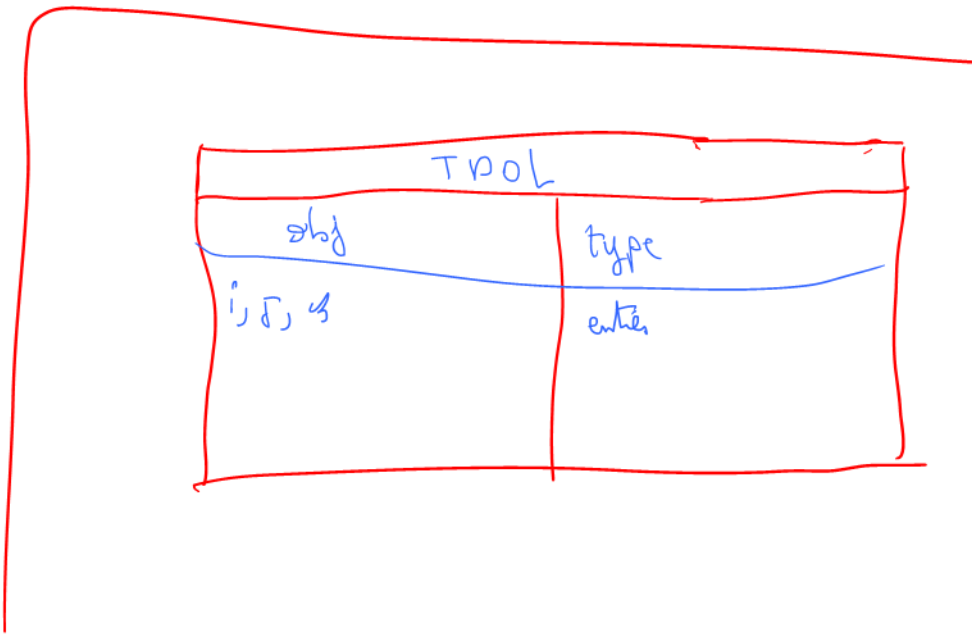
65 + 4 = E

65 + 5 = F

```
Function msgj( m°, mot1, msg ) :chaine
  Debut
    Res ← ""
    pour i de 0 a Long(msg)-1 faire
        a ← 0
        b ← 0
        qui ← Faux
        Si msg[i] ≠ " " alors
          tant que (a < 6 et b < 6) ou non qui faire
            Si msg[i] = m[a,b] alors
              Res ← Res + chr( ord("A") + b ) + chr( ord("A") + a )
              qui ← Vrai
            fi si
            b ← b + 1
            si b = 6 alors
              a ← a + 1
              b ← 0
          Fin tantque
          sinon
            Res ← Res + msg[i]
        Fin si
    Fin pour
    retourner Res
  Fin
```

TDO L	
objet	type
a, b, i	entier
Res	chaine

```
procedure FillM(m: mot 2, msg: chaine, solt: chaine)
  Debut
    pour k de 0 ou 5 faire
      m[0,k] ← solt[k]
    fin pour
    k ← 0
    i ← 1
    j ← 0
    tant que k < Long(msg) ou (i < 7 et j < 6) faire
      m[i,j] ← msg[k]
      k ← k + 1
      j ← j + 1
      si j = 6 alors
        i ← i+1
        j ← 0
      Fin si
    Fin tant que
  Fin
```



```
procedure sort (m : mot 2) :
  Debut
    // bubble !
    isSorted ← faux
    tant que non isSorted faire
      isSorted ← vrai
      pour i de 0 a 4 faire
        si ord(m[0,i]) > ord(m[0,i+1]) alors
          swapp ← m[0,i]
          m[0,i] ← m[0,i+1]
          m[0,i+1] ← swapp
          isSorted ← faux
        Fin si
      Fin pour
    Fin tantque.
  Fin
```

Objet	Type
isSorted	booleen
i	entier
swapp	chaine de caractère.

```
Fonction hashed (m : mot 2) : chaine
  Debut
    Res ← ""
    pour i de 0 à 6 faire
      pour j de 0 à 5 faire
        Res ← Res + m[i,j]
      Fin pour
    Fin pour
    retourner Res
  Fin
```

TODL	
Objet	type
i, j	entier
Res	chaine