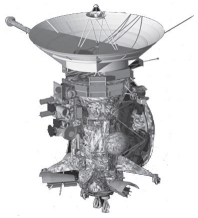
DS 4



La mission Cassini-Huygens

Concours CCINP – PSI 2016

1 Exercices

```
def factorielle(n) :
    """
    Donnée : un entier n >= 0
    Resultat : la factorielle de n
    """
    F=1
    i = n
    while i > 0 :
        F = F*i
        i = i-1
    return F
```

Question 1 *Montrer que $F \times i! = n!$ est un invariant de boucle.*

Correction
- À l'entrée dans la boucle, $F = 1$, $i = n$; donc $F \times i! = n!$.
- On considère qu'au k^{e}tour de boucle, $F_k \times i_k! = n!$.
- Montrons que $F_{k+1} \times i_{k+1}! = n!$. À la fin de l'itération suivante, calculons $F_{k+1} \times i_{k+1}! = (F_k \times i_k) \times (i_k - 1)! = F_k \times (i_k \times (i_k - 1)!) = F_k \times i_k! = n!$. *CQFD*

```
def racine(n):
    """
    Données : un entier n >= 0
    Résultat : la racine carrée de n (arrondie à l'
        entier inférieur)
    """
    c = 0
    s = 1
    while s <= n :
        c = c + 1
        s = s + 2*c + 1
    return c
```

Question 2 *Montrer que $n - s$ est un variant de boucle.*

Correction
- D'après la condition du while, la quanité $n - s$ est toujours positive.
- n ne varie pas et c est croissant est positif à chaque itération ; donc $n - s$ décroît à chaque itération.

$n - s$ est donc un variant de boucle.

2 Introduction

3 Acquisition d'images par l'imageur VIMS et compression des données

3.1 Présentation

Question 3 *Déterminer en octets la taille d'un cube de données hyperspectrales avant compression.*

Correction La taille d'une image est de $t_i = 64 \times 64 \times 12 = 49\,152$ bits soit, pour le cube une taille de $t_c = t_i \times 352 = 17\,301\,504$ bits $= 2\,162\,688$ octets $= 2{,}16$ Mo.

Question 4 *Déterminer alors le taux de compression à appliquer aux données afin de respecter le cahier des charges.*

Correction La taux de compression à appliquer est donc $\tau = \dfrac{T - T_c}{T} = \dfrac{2{,}16 - 1}{2{,}16} = 54\,\%$.

3.2 Principe de la compression des données

3.3 Limite du taux de compression

Question 5 *Calculer l'entropie associée à l'exemple précédent. En déduire le taux de compression limite de cet exemple et le comparer à la longueur moyenne de 2,4 bits par caractère.*

Correction $H(S_n) = -\sum_{i=1}^{i=6} p_i \log_2 p_i = -2 \times 0,3 \log_2 0,3 - 4 \times 0,1 \log_2 0,1 = 2,37.$

*****

Question 6 *Compléter le commentaire 1 de la fonction* `entropie(S)` *correspondant à la ligne de programme définissant la variable* `valeurs`.

Correction

Question 7 *Compléter la 2e étape de la fonction* `entropie(S)` *à partir du commentaire afin de calculer les probabilités* p_i.

Correction

Question 8 *Compléter la 3e étape de la fonction* `entropie(S)` *afin de calculer l'entropie de Shannon* H *définie par l'équation (1).*

Correction Questions 4, 5 et 6.

```
def entropie (S) :
    # Question 4
    """ Détermine l'entropie àpartir d'une liste d'entiers :
     * S (list[int])
     * H : entropie
    """
    valeurs = list(set(S))
    # Question 5
    prob = []
    for v in valeurs :
        nb_occ = 0
                for el in S :
                    if el == v :
                        nb_occ=nb_occ+1
        prob.append(nb_occ/len(S))

    # Question 6
    H = 0
    for p in prob :
        H=H-p*log2(p)

    return H
```

Suite à une acquisition d'image par l'instrument VIMS, l'utilisateur dispose de données brutes dont il souhaite évaluer l'entropie. Un bloc élémentaire de données se présente sous la forme d'un tableau à une dimension `bloc_image` constitué de valeurs entières (allant de 0 à $4095 = 2^{12} - 1$).

On rappelle que H est une valeur limite du taux de compression que l'on peut espérer atteindre.

Question 9 *Écrire la suite d'instructions permettant de calculer et d'afficher la valeur du taux de compression limite* τ *associé aux données contenues dans le tableau* `bloc_image`.

Correction

```
n = 4095
H = entropie(bloc_image)
tau=(n*12-n*H)/(n*12)
```

3.4 Prétraitement des données avant compression

3.5 Compression des données

3.5.1 Prédiction du mappage

Question 10 *Écrire une fonction* `prediction(x)` *recevant en argument d'entrée le tableau à une dimension* `x` *et renvoyant le tableau* `erreur` *contenant les valeurs* Δ_k.

Correction
```
def prediction(x):
    return [x[i]-x[i-1] for i in range(1,len(x))]# on pourrait mettre 32 àla place de len(x)
```

Question 11 *À partir de la définition de la fonction de mappage, équation (2), écrire une fonction* `mappage(erreur,x)` *recevant en arguments d'entrée les tableaux à une dimension* `erreur` *et* `x` *et renvoyant le tableau* `delta` *contenant les valeurs* δ_k.

Correction
```
def mappage(erreur,x):
    delta=[]
    for i in range(1,len(x)):
        theta=min([x[i-1],4095-x[i-1]])
        if erreur[i-1]<=theta and erreur[i-1]>=0:
            delta.append(2*erreur[i-1])
        elif erreur[i-1]<0 and erreur[i-1]>=-theta:
            delta.append(-2*erreur[i-1]-1)
        else:
            delta.append(theta-erreur[i-1])
    return delta
```


3.5.2 Codage entropique - Algorithme de Rice

Question 12 *Déterminer le code de Rice associé à la suite de valeurs* δ_k *données dans le* **??**, *page 7 pour* $p = 3$. *On pourra remplir le tableau suivant.*

Correction

δ_k	Quotient par $2^p = 8$	Codage unaire	Reste	Codage binaire	Codage Complet
4	0	0	4	100	0-100
10	1	10	2	010	10-010
18	2	110	2	010	110-010
18	2	110	2	010	110-010
6	0	0	6	110	0-110
1	0	0	1	001	0-001
3	0	0	3	011	0-011

Question 13 *Définir la suite d'instructions de la fonction* `codage(delta_k,p_opt)` *permettant d'obtenir un tableau à une ligne code1 associé à la première partie du code de Rice (codage unaire du quotient).*

Correction

Question 14 *Définir la suite d'instructions de la fonction* `codage(delta_k,p_opt)` *permettant d'obtenir un tableau à une ligne* `code2` *associé à la seconde partie du code de Rice (codage binaire du reste) et réalisant ensuite l'assemblage des deux parties dans le tableau code. L'instruction* `bin` *de Python pourra être utilisée. Elle permet de convertir nu entier en binaire (on supposera que cette fonction renvoie une chaîne de caractères associée au code binaire).*

Correction
```
def codage(delta_k,p_opt):#résultat sous forme d'un tableau
    #Q15
    quotient=delta_k//2**p_opt
    code1=[1 for i in range(quotient)]
    code1+=[0]
    #Q16
    reste=delta_k%2**p_opt#codable sur p_opt bits
    code2=[int(i) for i in bin(reste)[2:]]
    for i in range(p_opt-len(code2)):
        code2=[0]+code2
    code=code1+code2
    return code
```