Montre multisport
Corrigé

Question 1 Expliquer les lignes 2 à 9 de cette fonction.

#2 : on découpe chaine sur les ' \ n '

#3 : on récupère l'indice de la ligne avec GPGGA

#4 : on initialise ligne comme liste vide

#5 : si l'indice de la ligne trouvée est différent de la longueur de chaine De coupe

#6 : alors on récupère la ligne en supprimant le 1er caractère $

#7 : ou coupe la ligne à chaque virgule

#8 : on supprime l'avant dernier terme toujours vide

#9 : on renvoie la liste crée (éventuellement vide)

Question 2 Écrire une fonction indiceGPGGA(listeChaine) prenant en argument une liste de chaines de caractères et qui renvoie l'indice de la position de la chaine contenant '$GPGGA' quand elle existe ou la longueur de listeChaine sinon. Vous utiliserez une boucle while.

```
def indiceGPGGA (listeChaine):
    i=0
    while i < len(listeChaine) and '$GPGGA' not in listeChaine [i]:
        i = i +1
    return i
```

ou, pour le test: **while** i<len(listeChaine) **and** listeChaine [:6] !='$GPGGA':

Question 3 Donner l'instruction à écrire pour récupérer la trame GPGGA de la variable donneesGPS et la stocker dans la variable listeGPGGABrute.

```
ListeGPGGABrute = recupererGPGGA (donneeGPS)
```

Question 4 Écrire une fonction testPresence(trame) qui prend en argument la liste de chaînes de caractères trame correspondant à une trame GPGGA et qui renvoie True si les données sont présentes et non vides, False sinon.

```
def testPresence (trame ) :
    return not  in trame and len (trame) ==14
```

Sans l'utilisation de l'opérateur in :

```
def testPresence(trame):
    if len(trame) < 14: return False
```

```
    for i in range(14):
        if trame[i]=='': return False
    return True
```

Question 5 Écrire une fonction testPrecision(trame) qui prend en argument la liste de chaînes de caractères trame correspondant à une trame GPGGA et qui renvoie True si la précision est inférieure à 5, False sinon.

```
def testPrecision( trame ):
    return float (trame [ 8 ] ) <5
```

Question 6 Donner la valeur renvoyée par la fonction testSC appliquée à la liste ['GPG','*A0'].

Donner notamment la valeur de la variable csHex. On donne les résultats suivants :

bin(ord('G')) renvoie '0b01000111', bin(ord('P')) renvoie '0b01010000',

bin(ord('*')) renvoie '0b00101010', bin(ord('A')) renvoie '0b01000001',

bin(ord('0')) renvoie '0b00110000'.

La fonction calcule le Ou Exclusif sur les représentations binaires de 'GPG' uniquement,

soit 01000111^ 01010000 ^ 01000111 = 01010000 = 50 en hexadécimal.

La fonction renvoie donc False car '50' != 'A0'.

Question 7 Écrire une fonction convHoraire(chHoraire) qui prend en argument la chaîne de caractères chHoraire représentant l'horaire avec le format de la norme GPGGA (092828.00 correspondant à 09h 28min 28,00 secondes) et qui renvoie la valeur de l'horaire exprimé en secondes.

```
def convHoraire ( chHoraire ) :
    h = int ( chHoraire [ : 2 ] )
    m = int ( chHoraire [ 2 : 4 ] )
    s  = float ( chHoraire [ 4 : ] )
    return 3600* h+60*m+s
```

Question 8 Écrire une fonction convAngle(chAngle,chCard) qui prend en argument la chaîne de caractères chAngle correspondant à la latitude ou la longitude ainsi que la chaîne de caractères chCard correspondant à la direction (N/S ou E/O) et qui renvoie le flottant signé correspondant à la valeur de l'angle en degré (il faudra convertir les minutes d'angle en valeur décimale).

```
def convAngle (chAngle, chCard):
    Angle=float (chAngle [: -8])+ float (chAngle [-8:]) / 60
    if (chCard=='E' or chCard=='S'):
        return -Angle
```

```
    return Angle
```

Alternative pour la première ligne :

```
n = len(chAngle)
Angle = float(chAngle[ :n-8]) + float(change[n-8:])/60
```

Question 9 Compléter l'ébauche de la fonction recupDonnees(trame) qui prend en argument trame correspondant à une trame GPGGA et qui renvoie une liste de 4 réels contenant (Heure, Latitude, Longitude, Altitude).

```
def recupDonne ( trame ) :
    Horaire = convHoraire (trame [1])
    Lat = convAngle (trame [2], trame [3])
    Long = convAngle (trame [4], trame [5])
    Alt = float (trame [9])
    return Horaire, Lat, Long, Alt
```

Question 10 On suppose que l'on réalise une activité d'une heure. Donner l'ordre de grandeur de la taille mémoire en octets du fichier texte généré. Dans le cahier des charges, on impose que la montre peut sauvegarder 200 h d'activités. Donner l'ordre de grandeur de la taille mémoire nécessaire au stockage des données pour répondre au cahier des charges.

La chaîne de caractère est composée de 38 caractères. Chaque caractère étant codé sur 1 octet, une ligne représente 38octets. En 1 heure on aura 3600 enregistrements donc la taille du fichier sera de 3600 X 38 = 136800octets = 134kio

Pour sauvegarder 200h d'activités, il faudrait donc une mémoire supérieure à 26Mio (27,4 Mo).

Question 11 Proposer une solution permettant de ne pas dépasser la mémoire disponible.

Pour les activités lentes, il n'est pas nécessaire de sauvegarder les informations toutes les secondes.

Cela permettrait facilement de respecter le cahier des charges.

Question 12 Montrer qu'on obtient, après discrétisation et en appliquant la méthode des trapèzes pour calculer une valeur approchée de l'intégrale dans l'équation, la relation de récurrence suivante

$$S_{k+1} = \frac{(1-G)S_k + G.(e_{k+1}+e_k)}{1+G} \qquad (1)$$

où G est une constante dont vous donnerez l'expression en fonction des paramètres fc et fe.

En appliquant la formule des trapèzes sur l'expression, on obtient :

$$S_{k+1} = S_k + 2\pi f_c \frac{(e_k - s_k) + (e_{k+1} - s_{k+1})}{2} T_e$$

or Te = 1/ fe

Après calcul, on obtient la forme demandée avec G = π. fc / fe

Question 13 Écrire la fonction filtrage(e,G) d'argument e tableau du signal d'entrée, G la constante définie précédemment et qui renvoie le tableau de valeurs du signal filtré.

```
def filtrage (e, G) :
    s =[ e [ 0 ] ]                    # initialisation de la sortie filtrée
    for i in range (len (e) - 1) :
        s.append ( ( (1 -G) * s [ i ]+G*( e [ i ]+e [ i +1] ) ) / ( 1+G) )
    return s
```

Question 14 Compléter le test du while de la fonction detectionPics(extSignal,seuil) qui prend en argument extSignal, liste des données du signal partiel avec 150 points de mesure ainsi que la valeur minimale seuil et qui renvoie l'indice du premier « pic » dans la partie extSigna du signal traité.

```
def detectionPics (ext Signal , seuil) :
    P0 , P1 , P2 = extSignal [ : 3 ]
    i =1
    while i <len ( extSignal ) - 2 and ( P0>P1 or P1<P2 or P1<seuil ) :
        P0 , P1 = P1 , P2
        P2=extSignal [ i +2]
        i = i +1
    return i
```

Question 15 Commenter précisément la fonction pulsationCardiaque(signal,Te,seuil) qui prend en argument signal, liste des données du signal complet, Te le temps d'échantillonnage en seconde et seuil la valeur du seuil de détection. Expliciter ce que représente la valeur renvoyée.

#2 : initialisation de la liste des pics

#3 : initialisation à 0 de l'indice de début de la recherche

#4 : nombre de points correspondant à 3s pour la recherche sur une fenêtre

#5 : tant qu'il est possible d'analyser une fenêtre

#6 : on détecte l'indice d'un nouveau pic

#7 : que l'on ajoute à la liste des pics

#8 : on calcule la moyenne des battements en calculant le temps entre le premier et le dernier battement divisé par le nombre d'intervalle entre deux pics

#9 : on transforme en battements par minutes.

Question 16 Écrire une fonction TFD (signal,Te) qui prend en argument signal, liste des données du signal et Te le temps d'échantillonnage et qui renvoie la liste des fréquences et la liste des module de jS (k)j. La fonction abs(z) renvoie la valeur du module du complexe z (fonctionne sur un tableau) ; le complexe 1+2i s'écrit en Python 1+2j; le complexe a+ib s'écrit en Python a+b*1j.

```
def TFD( signal , Te ) :
    N = len (signal)
    S, F = [ ], [ ]
    for k in range ( 0 ,N) :
        s = 0
        for n in range ( 0 ,N) :
            s += signal [ n ] * exp (-2 j * pi *k*n /N)
        S. append ( abs ( s ) )
        F.append ( k / N / Te)
    return F , S
```

Question 17 Justifier la présence d'une valeur très élevée pour une fréquence nulle.

Le signal oscille autour d'une valeur constante non nulle.

Question 18 Modifier la fonction TFD précédente pour qu'elle ne calcule que les valeurs de S (k) comprises entre ces deux fréquences.

Il ne faut prendre dans le vecteur k que les valeurs qui nous intéresse, valeurs définies par les 3 lignes après la définition de k :

```
def TFD( signal , Te ) :
    N = len (signal)
    S = [ ]
    Kmin = int (N*0.5* Te)
    Kmax = int (N*4*Te)
    for k in range ( kmin , kmax+1) :
        s = 0
```

```
        for n in range ( 0 ,N) :
            s += signal [ n ] * exp (-2 j * pi *k*n /N)
    S.append ( abs ( s ) )
    return arange (N) [ kmin : kmax +1 ] /N/ Te , S
```

Question 19 Donner l'intervalle Δt choisi par le programmeur. Expliquer la ligne 69 et compléter les lignes 70 et 71.

Le programmeur a choisi un Δt de 600 points soit 600 * 0.02 = 12 s.

A la ligne 69, le programme calcul la fréquence cardiaque en prenant la fréquence maximale dans la TFD (freq[Sk.index(max(Sk))]). On obtient le nombre de battement par secondes, que l'on multiplie par 60 pour avoir la fréquence cardiaque en minutes.

```
70  indDepart = indDepart+Fe      # il faut décaler d'une seconde donc Fe
71  indFin = indFin+Fe
```

Question 20 Entourer sur le document réponse la fonction correspondant à cette définition.

La fonction calculée est $\frac{1}{2}\left(1-\cos\left(\frac{2\pi}{\Delta t}t\right)\right)$, toujours positive, valant 0 pour t=0.

Il s'agit de la courbe en haut à droite.

Question 21 Expliquer comment modifier le programme donné à la question Q19 pour prendre en compte la fenêtre de Hann précédente.

Il suffit d'appliquer la fonction Hann sur la partie du signal à analyser.

```
partSignal = Hann ( partSignal , Fe )
```

Question 22 Écrire une requête SQL permettant de récupérer la liste des identifiants des activités du membre dont l'identifiant est 1.

```
SELECT Ida FROM activite WHERE idm=1 ;
```

Question 23 Écrire une requête SQL permettant de donner la date, la distance parcourue et la vitesse moyenne en km/h des activités de type « course » du membre dont l'identifiant est 1.

```
SELECT date, Distance, Distance/Temps*3600 FROM activite WHERE idm=1 and Type='course' ;
```

Question 24 Décrire chaque instruction de la requête et expliquer ce qu'elle renvoie.

La requête : SELECT membre1 AS idam1 FROM 'amis' WHERE membre2=1

Crée la table des membres dont le membre2 est 1.

La requête :

```
SELECT membre2 AS idam1 FROM amis WHERE membre1=1
```

Fait la même chose avec le membre 1

Et on fusionne (avec UNION) les deux tables pour obtenir la table amis1 contenant tous les amis du membre dont l'identifiant est 1.

On réalise alors une jonction entre cette table et la table activite pour obtenir la liste des activités de marche des membres qui sont amis du membre 1.