Question n° 1 : Programme complet -- 6 pts

SuperSauteur a super besoin de super beaucoup de temps pour super-sauter super loin et super vite.

Le jour de son premier anniversaire il pouvait super-sauter en une seconde à : 1^2 centimètre.

Le jour de son second anniversaire il pouvait super-sauter en une seconde à $1^2 + 2^2$ centimètres.

Le jour de son troisième anniversaire il pouvait super-sauter en une seconde à $1^2 + 2^2 + 3^2$ centimètres.

Le jour de son quatrième anniversaire il pouvait super-sauter en une seconde à $1^2 + 2^2 + 3^2 + 4^2$ centimètres.

etc.

On souhaite écrire une fonction `calculer_longueur_saut` qui prend en paramètre un entier positif n et retourne le nombre entier correspondant à la longueur de saut de SuperSauteur le jour de son n ième anniversaire.

1. Donner le prototype de cette fonction.
2. Faut-il utiliser une boucle bornée ou une boucle non-bornée ? Justifier la réponse.
3. Noémie a étudié les variables dont elle pense avoir besoin pour coder cette fonction en langage Python. Elle a noté sur son brouillon :
   - `n` : âge de SuperSauteur
   - `anniversaire` va parcourir 1, 2, 3, ... n
   - `longueur_saut = longueur_saut + anniversaire ** 2`

   Son enseignant lui dit : "c'est un très bon début, il n' y a plus qu'à formaliser tout ça"

   En utilisant les mêmes noms de variables que Noémie, écrire le corps de la fonction `calculer_longueur_saut(n)`
4. Ecrire le code d'une seconde fonction `calculer_age_mur_du_son` qui ne prend aucun paramètre mais retourne l'âge à partir duquel SuperSauteur dépassera la vitesse du son lors de ses sauts (c'est-à dire fera des sauts d'une longueur supérieure à 34 000 centimètres en une seconde)

Question n° 2 : Programme complet -- 6 pts

Depuis ses huit ans, SuperAvare économise des kopecks (monnaie locale).

Le jour de ses huit ans il avait 8^3=512 kopecks dans sa tirelire.

Le jour de ses neuf ans il avait 512+9^3=1241 kopecks dans sa tirelire.

Le jour de ses dix ans il avait 1241+10^3=2241 kopecks dans sa tirelire.

Le jour de ses onze ans il avait 2241+11^3=3572 kopecks dans sa tirelire.

etc.

On souhaite écrire une fonction `calculer_tirelire` qui prend en paramètre un entier positif n et retourne le nombre entier correspondant au montant de la tirelire de SuperAvare le jour de son n ième anniversaire.

1. Donner le prototype de la fonction calculer_tirelire
2. Faut-il utiliser une boucle bornée ou une boucle non-bornée ? Justifier la réponse.
3. Lucie a étudié les variables dont elle pense avoir besoin pour coder cette fonction en langage python. Elle a noté sur son brouillon : - n = âge de SuperAvare - anniversaire = 8, 9, 10, 11, …n - somme_tirelire = somme_tirelire + anniversaire**3 Son enseignante lui a dit : "c'est un très bon début, en plus c'est assez clair".

   - `n` : âge de SuperAvare
   - `anniversaire` va parcourir 8, 9, 10, 11 ... n
   - `somme_tirelire = somme_tirelire + anniversaire ** 3`

   Son enseignante lui dit : "c'est un très bon début, il n' y a plus qu'à formaliser tout ça"

   En utilisant les mêmes noms de variables que Lucie, écrire le corps de la fonction `calculer_tirelire(n)`
4. écrire le code d'une seconde fonction `age_millionaire()` qui ne prend aucun paramètre mais retourne l'âge à partir duquel SuperAvare sera kopeck-millionnaire.

Question n° 3 : Boucle while -- 3 pts

Quelle est la valeur de a à la fin de l'exécution du programme ci-dessous ?

```
r = 0
while b != 0:
    r = a % b
    a = b
    b = r
```

1. lorsque a = 10 et b = 3
2. lorsque a = 10 et b = 15
3. Que permet de calculer ce programme ?

Question n° 4 : Boucle while -- 3 pts

Quelle est la valeur de a à la fin de l'exécution du programme ci-dessous ?

```
r = 0
while b != 0:
    r = a % b
    a = b
    b = r
```

1. lorsque a = 6 et b = 15
2. lorsque a = 6 et b = 5
3. Que permet de calculer ce programme ?

Question n° 5 : Boucle while -- 3 pts

Quelle est la valeur de a à la fin de l'exécution du programme ci-dessous ?

```
r = 0
while b != 0:
    r = a % b
    a = b
    b = r
```

1. lorsque a = 15 et b = 21
2. lorsque a = 15 et b = 7
3. Que permet de calculer ce programme ?

Question n° 6 : Corriger un programme -- 2 pts

On veut calculer la somme des inverses des n premiers entiers.

Le programme suivant comporte des erreurs.

Indiquer les types d'erreur et les corriger.

Proposer une version complète du programme corrigé.

```
def somme_inv_entiers(n)
    for i in range(n):
        s = s + 1/i
    return i
```

Question n° 7 : Corriger un programme -- 2 pts

On veut calculer le produit des entiers de 1 à n (n est étant un entier supérieur ou égal à 1).

Le programme suivant comporte des erreurs.

Indiquer les types d'erreur et les corriger.

Proposer une version complète du programme corrigé.

```
def produit_entiers(n):
    p = 0
    for i in range(n)
        p = p * n
    return n
```

Question n° 8 : Corriger un programme -- 2 pts

On veut écrire une fonction qui détermine si un entier est un nombre premier.

Le programme suivant comporte des erreurs.

Indiquer les types d'erreur et les corriger.

Proposer une version complète du programme corrigé.

```
def est_premier():
    if nombre <= 1:
        return False
    diviseur = 2
    while diviseur < nombre
        if nombre % diviseur = 0:
            return True
    return False
```

Question n° 9 : Tableau exécution -- 3 pts

Soit le programme Python suivant :

```
cpt = 0
for i in range(4):
    for j in range(i):
        if (i + j) % 2 == 0:
            cpt = cpt + 1
```

Remplir le tableau d'exécution suivant (en utilisant le nombre de lignes nécessaires)

i	j	cpt

Question n° 10 : Tableau exécution -- 3 pts

Soit le programme Python suivant :

```
cpt = 0
for i in range(2,5):
    for j in range(i-1):
        if (i + j) % 2 == 1:
            cpt = cpt + 1
```

Remplir le tableau d'exécution suivant (en utilisant le nombre de lignes nécessaires)

i	j	cpt

Question n° 11 : Linux -- 4 pts

"Linux ou GNU/Linux est une famille de systèmes d'exploitation open source de type Unix fondée sur le noyau Linux, créé en 1991 par Linus Torvalds. De nombreuses distributions Linux ont depuis vu le jour et constituent un important vecteur de popularisation du mouvement du logiciel libre."

Source : Wikipédia, extrait de l'article consacré à GNU/Linux.

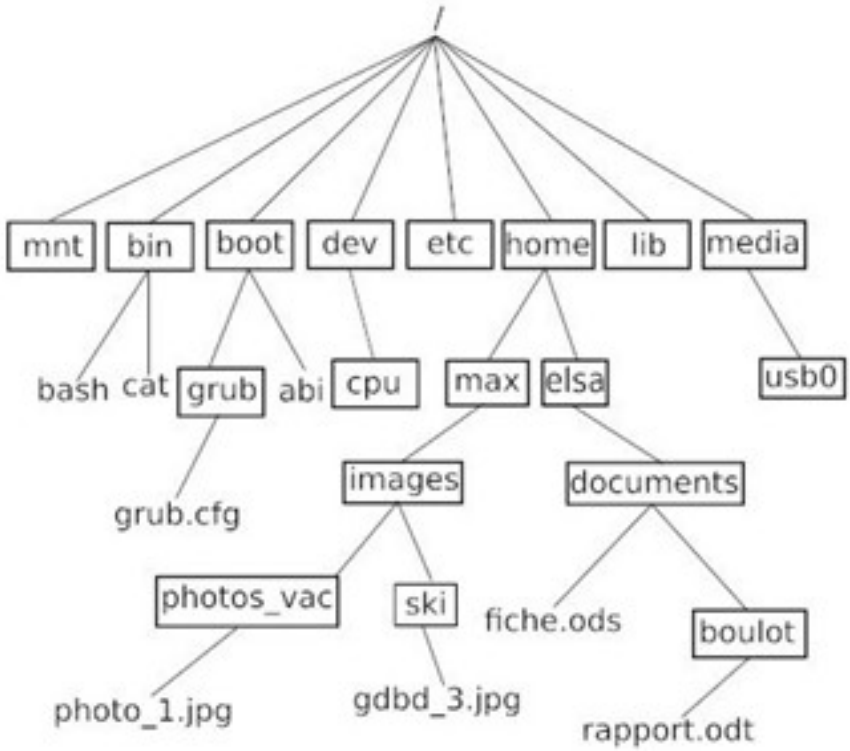
"Windows est au départ une interface graphique unifiée produite par Microsoft, qui est devenue ensuite une gamme de systèmes d'exploitation à part entière, principalement destinés aux ordinateurs compatibles PC. Windows est un système d'exploitation propriétaire."

Source : Wikipédia, extrait de l'article consacré à Windows.

1. Expliquer succinctement les différences entre les logiciels libres et les logiciels propriétaires.

2. Expliquer le rôle d'un système d'exploitation.

On donne ci-dessous une arborescence de fichiers sur un système GNU/Linux (les noms encadrés représentent des répertoires, les noms non encadrés représentent des fichiers, / correspond à la racine du système de fichiers) :



3. Indiquer le chemin absolu du fichier *rapport.odt*.

On suppose que le répertoire courant est le répertoire *elsa*.

4. Indiquer le chemin relatif du fichier *photo_1.jpg*.

L'utilisatrice Elsa ouvre une console (aussi parfois appelée terminal), le répertoire courant étant toujours le répertoire *elsa*.

On fournit ci-dessous un extrait du manuel de la commande UNIX *cp* :

```
NOM
  cp - copie un fichier
UTILISATION
  cp fichier_source fichier_destination
```

5. Déterminer le contenu du répertoire *documents* et du répertoire *boulot* après avoir exécuté la commande suivante dans la console :

```
cp documents/fiche.ods documents/boulot
```

Question n° 12 : Tests et spécification de fonctions -- 2 pts

Reprendre le code de ce programme et y ajoutant :

- la description de ce que fait cette fonction à l'aide des *docstrings*
- 2 tests au format *doctest*
- Deux instructions *assert* testant la fonction dont une des deux doit provoquer une erreur.

```
def mystere(chaine,c):
    n = 0
    for l in chaine:
        if l == c:
            n = n + 1
    return n
```

Question n° 13 : Tests et spécification de fonctions -- 2 pts

Reprendre le code de ce programme et y ajoutant :

- la description de ce que fait cette fonction à l'aide des *docstrings*
- 2 tests au format *doctest*
- Deux instructions *assert* testant la fonction dont une des deux doit provoquer une erreur.

```
def mystere(chaine,c):
    for x in chaine:
        if c == x:
            return True
    return False
```