Devoir d'informatique n° 1 (2 heures)

Ce devoir est constitué de plusieurs petits exercices. L'ordre des exercices ne correspond à aucun critère de difficulté ou de longueur : vous pouvez les traiter dans l'ordre que vous voulez. Veillez à soigner la copie tant pour l'écriture, la propreté que pour la rédaction, la rigueur et l'argumentation. De plus, on prêtera une attention particuliére au respect des alignements et des indentations des séquence d'instructions Python. La calculatrice est autorisée.
Vous numéroterez vos copies et ferez apparaître clairement sur la première page le nombre de copies.

Exercice 1

On rappelle qu'en base 8, on utilise les chiffres de 0 à 7, et qu'en base 16, on utilise les chiffres usuels (de 0 à 9) ainsi que les "chiffres" A, B, C, D, E et F qui représentent respectivement 10, 11, ...15.

1. Convertir en binaire, en octal (i.e. en base 8), en hexadécimal (base 16), les nombres n, m et p suivants (qui sont écrits en base 10) : $n = 1532$, $m = 2864$ et $p = 4037$.
2. Convertir en décimal les nombres suivants : 1011011101_{bin}, 2537_{oct} et $2C5E_{hex}$.
3. Calculer la somme et le produit des nombres binaires 11001101 et 1110101. On fera apparaitre les calculs sur la copie et on restera en binaire ...
4. Calculer (en faisant apparaitre les calculs sur la copie et en restant dans la base indiquée) : $6B9_{hex} + D7_{hex}$, $1150_{oct} - 562_{oct}$ et $1150_{oct} \times 562_{oct}$.

Exercice 2

On suppose que notre machine numérique écrit les nombres entiers en binaire sur 8 bits uniquement.

1. Donner les algorithmes (écrits en français) pour calculer $N \times 4$, $N \mod 8$ et $N//2$ (où // est la division entière). Ces algorithmes ne doivent faire intervenir aucune opération arithmétique. Indiquer pour chacun, le nombre de bits qui sont nécessaires pour le résultat de chaque calcul.
2. Reproduire et compléter le tableau suivant

N écrit en base 10	N écrit en binaire	$N \times 4$ écrit en binaire	$N//2$ écrit en binaire	$N \mod 8$ écrit en binaire
12	00001100			
54	00110110			
74	01001010			

Exercice 3

Quelles sont les valeurs finales de a, b, c et d à la fin des instructions suivantes ?

```
>>> a = 18
>>> b = 4
>>> c = a/b
>>> d = a%b
>>> a = a//b
>>> b = a//b
```

Exercice 4

1. Quel est l'écriture décimale du nombre réel dont l'écriture binaire est 10101.1011 où le "." représente la "virgule binaire") ?
2. Donner l'écriture binaire (infinie périodique ...) du nombre décimal $0,3$ (ici la "," est la virgule décimale). En déduire l'écriture binaire du nombre $2560,3$.
3. Si on ne retient que les 16 premiers chiffres de la décomposition binaire de $0,3$, quel est le nombre décimal exact ainsi représenté ? Même question avec $2560,3$.

Exercice 5

Ecrire un algorithme (en français **puis** en Python) permettant d'afficher, dans l'ordre demandé, les $2n^4$ pour les entiers n multiples de 7 décroissants de 49 à 14.

Exercice 6 En France, le calcul de l'impôt sur le revenu suit un barème progressif : le taux "marginal" (la définition de ce terme n'est pas nécessaire ici) croit avec les revenus. Si un foyer fiscal possède un revenu imposable R et un nombre de parts N (ce qui correspond grosso modo au nombre de personnes constituant le foyer fiscal) , on calcule son "quotient familial" par la formule $\mathrm{QF} = \frac{\mathrm{R}}{\mathrm{N}}$. Le montant de l'impôt pour ce foyer est alors calculé selon le barème suivant

Si QF est	supérieur ou égal à	et strictement inférieur à	alors l'impôt dû est
	0	6 011	0
	6 011	11 991	$R \times 0,055 - N \times 330$
	11 991	26 631	$R \times 0,14 - N \times 1\,349$
	26 631	71 397	$R \times 0,30 - N \times 5\,610$
	71 397	151 200	$R \times 0,41 - N \times 13\,463$
	151 200		$R \times 0,45 - N \times 19\,511$

1. Ecrire une séquence d'instructions Python qui, supposant connues les valeurs de R et de N, calcule et affiche le montant de l'impôt dû
2. Modifier cette séquence d'instructions, en tenant compte du fait que l'administration fiscale ne tient pas compte des centimes.

Exercice 7

On considère la fonction appelée "fonction1" suivante :

```
>>> def fonction1(a1,b1):
>>>     a = floor(a1)       #floor est la fonction partie entiere dans le module math
>>>     s = 1
>>>     while a > 1:
>>>         s = s * b1**2
>>>         a = a-2
>>>         print(a,s)
>>>     if a == 1:
>>>         return(s * b1)
>>>     else:
>>>         return (s)
```

1. Quels sont les affichages successifs et le retour de l'appel " fonction1(4.6, 5)"
2. Quels sont les affichages successifs et le retour de l'appel " fonction1(5, 4)"
3. On suppose qu'on applique la fonction1 à un couple (a1,b1) de nombres flottants positifs. Montrer que l'appel fonction1(a1,b1) fournit bien une réponse et que la boucle while se termine effectivement.
4. On suppose que l'on efface la ligne 7 du script de la fonction1 (l'instruction print(a,s)). On suppose également qu'on applique la fonction1 à un couple d'entiers naturels (a1,b1). Quel est le résultat retourné par l'appel fonction1(a1,b1) ?

Exercice 8

Ecrire un algorithme (en français **puis** en Python) qui, dans une liste L de nombres, affecte aux variables **V1**, **V2** et **V3**, le nombre d'éléments de la liste L compris strictement entre 5 et 10 (pour **V1**), la somme des éléments de L qui sont multiples de 7 (pour **V2**) et le produit des éléments de L qui sont supérieurs ou égaux à 5 (pour **V3**).

Exercice 9 Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 1$ et $\forall n \in \mathbb{N}, u_{n+1} = \frac{2*u_n+9}{u_n+2}$

1. Ecrire une séquence d'instructions Python permettant de calculer une valeur approchée de u_{20}
2. On admet que la suite converge et que sa limite se trouve toujours entre deux termes consécutifs de la suite.
   Ecrire une séquence d'instructions Python de déterminer le plus petit entier n pour lequel u_n est une valeur approchée à 10^{-6} près, ainsi que cette valeur approchée.

CORRECTION

Exercice 1 : Décomposition des entiers en une base quelconque

On rappelle qu'en base 8, on utilise les chiffres de 0 à 7, et qu'en base 16, on utilise les chiffres usuels (de 0 à 9) ainsi que les "chiffres" A, B, C, D, E et F qui représentent respectivement 10, 11, ...15.

1. $1532 = 10111111100_{bin} = 10\,111\,111\,100_{bin} = 2774_{oct} = 101\,1111\,1100_{bin} = 5FC_{hex}$
   $2864 = 101100110000_{bin} = 5460_{oct} = B30_{hex}$
   $4037 = 111111000101_{bin} = 7705_{oct} = FC5_{hex}$

2. $1011011101_{bin} = 733$, $2537_{oct} = 1375$ **et** $2C5E_{hex} = 11358$.

3. $11001101 + 1110101 = 101000010$ **et** $11001101 \times 1110101 = 101110110110001$.

4. $6B9_{hex} + D7_{hex} = 790_{hex}$, $1150_{octal} - 562_{octal} = 366_{octal}$ et $1150_{octal} \times 562_{octal} = 675120_{octal}$.

Exercice 2

On suppose que notre machine numérique écrit les nombres entiers en binaire sur 8 bits uniquement.

1. $N \times 4$ **correspond à l'opération "décaler de deux crans vers la gauche"** ,
   $N \mod 8$ **correspond à l'opération "ne garder que les 3 derniers bits"** et
   $N//2$ **à l'opération "décaler d'un cran vers la droite"**

2. Reproduire et compléter le tableau suivant

N écrit en base 10	N écrit en binaire	$N \times 4$ écrit en binaire	$N//2$ écrit en binaire	$N \mod 8$ écrit en binaire
12	00001100	**00110000**	**00000110**	**00000100**
54	00110110	**11011000**	**00011011**	**00000110**
74	01001010	**00101000**	**00100101**	**00000010**

Remarquons que si on reste sur 8 bits, le calcul de 74×4 donne un résultat erroné

Exercice 3

Quelles sont les valeurs finales de a, b, c et d à la fin des instructions suivantes ? Après les instructions : a = 18 ; b = 4 ; c = a/b ; d = a%b ; a = a//b ; b = a//b ; on trouve **a = 4 , b = 1 , c = 4.5, d = 2**

Exercice 4

1. $10101.1011_{bin} = \frac{347}{16} = 21,6875$

2. $0,3 = 0.01\,0011\,0011\,\underline{0011}\ldots$ et donc $2560,3 = 101000000000.01\,0011\,0011\,\underline{0011}\ldots$

3. Si on ne retient que les 16 premiers chiffres de la décomposition binaire,
   on assimile $0,3$ **à** $0.0100110011001100_{bin} = \frac{4915}{16384} = 0,29998779296875$ exactement.
   De même, **on assimile** $2560,3$ **à** $101000000000.0100_{bin} = 2560,25$

Exercice 5

Algorithme permettant d'afficher les $2n^4$ pour les entiers n multiples de 7 décroissants de 49 à 14.

```
>>> for n in range(49,7,-7):
>>>         print(2 * n**4)
```

Exercice 6 On répond aux deux questions directement : pour la question 1, il suffit d'omettre la fonction floor

```
>>> def impot(R,N):
>>>         from math import floor
>>>         QF = R/N
>>>         if QF < 6011 :
>>>                         return(0)
>>>         elif QF < 11991 :
>>>                         return(floor(R*0.055 - N*330))
>>>         elif QF < 26631 :
>>>                         return(floor(R*0.14 - N*1349))
>>>         elif QF < 71397 :
>>>                         return(floor(R*0.30 - N*5610))
>>>         elif QF < 151200 :
>>>                         return(floor(R*0.41 - N*13463))
>>>         else :
>>>                         return(floor(R*0.45 - N*19511))
```

Exercice 7

On considère la fonction appelée "fonction1" suivante :

```
>>> def fonction1(a1,b1):
>>>     a = floor(a1)       #floor est la fonction partie entiere dans le module math
>>>     s = 1
>>>     while a > 1:
>>>         s = s * b1**2
>>>         a = a-2
>>>         print(a,s)
>>>     if a == 1:
>>>         return(s * b1)
>>>     else:
>>>         return (s)
```

1. Quels sont les affichages successifs et le retour de l'appel " fonction1(4.6, 5)" 2 25 0 625 625
2. Quels sont les affichages successifs et le retour de l'appel " fonction1(5, 4)" 3 16 1 256 1024
3. a valant initialement floor(a1), c'est un entier positif et on peut le comparer à 1. S'il est inférieur, le retour de la fonction est soit b1 soit 1. Sinon, on effectue des transformations et des affichages sur les variables a et s. A chaque boucle a est décrémenté de 2 donc il finira bien par devenir inférieur ou égal à 1, condition pour sortir de la boucle while. Aussi :
   l'appel fonction1(a1,b1) fournit bien une réponse et la boucle while se termine.
4. On considère l'invariant de boucle suivant : "à la fin de l'étape k de la boucle, s est affectée de la valeur numérique : $b1^{2^k}$ et a de la valeur : $a1 - 2k$". Il est aisé de constater qu'il s'agit bien d'un invariant de boucle. En sortie de boucle, a est affectée de la valeur numérique : $a1 - 2\left\lfloor \frac{a1}{2} \right\rfloor$ et s de la valeur $b1^{2\left\lfloor \frac{a1}{2} \right\rfloor}$. Après le test final, on se rend compte que
   l'appel fonction1(a1,b1) retourne donc la valeur $b1^{a1}$

Exercice 8

Ecrire un algorithme (en français **puis** en Python) qui, dans une liste L de nombres, affecte aux variables **V1**, **V2** et **V3**, le nombre d'éléments de la liste L compris strictement entre 5 et 10 (pour **V1**), la somme des éléments de L qui sont multiples de 7 (pour **V2**) et le produit des éléments de L qui sont supérieurs ou égaux à 5 (pour **V3**).

```
>>> V1, V2, V3 = 0, 0, 1
>>> for x in L:
>>>         if x>5 and x<10:
>>>                 V1 += 1
>>>         if x%7 == 0:
>>>                 V2 += x
>>>         if x >= 5:
>>>                 V3 *= x
```

Exercice 9 Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 1$ et $\forall n \in \mathbb{N}, u_{n+1} = \frac{2*u_n+9}{u_n+2}$

1. Ecrire une séquence d'instructions Python permettant de calculer une valeur approchée de u_{20}

```
>>> u = 1
>>> for k in range(20):
>>>         u = (2*u + 9)/(u + 2)
>>> print(u)
```

2. On admet que la suite converge et que sa limite se trouve toujours entre deux termes consécutifs de la suite.
   Ecrire une séquence d'instructions Python de déterminer le plus petit entier n pour lequel u_n est une valeur approchée à 10^{-6} près, ainsi que cette valeur approchée.

```
>>> u, v, n = 1, 11/3, 1
>>> while abs(v-u) < 10^-6:
>>>         z = v
>>>         v = (2*v + 9)/(v + 2)
>>>         u = z
>>>         n += 1
>>> print(n,v)
```