

Interrogation d'ITC n°4

Semaine du 09/06/2025

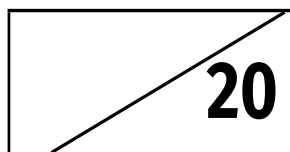
Durée : 25 minutes

Nom :

Prénom :

Consignes

- Les codes doivent être présentés en mentionnant explicitement l'indentation, au moyen par exemple de barres verticales sur la gauche.
- Pour qu'un code soit compréhensible, il convient de choisir des noms de variables les plus explicites possibles.
- La performance du code proposé à chaque question entrera dans l'évaluation, de même que l'utilisation de boucles appropriées.
- Vous pouvez bien sûr utiliser une feuille de brouillon en parallèle.



Exercice 1 Utilisation de la représentation des entiers positifs

1.  Donner l'ensemble des entiers naturels représentables sur 8 bits.

2.  Donner la représentation sur 8 bits de l'entier 57.

3.  Écrire une fonction `pair(L:list) -> bool` d'argument `L` une liste de 0 et 1 (on testera cette condition avec `assert`) et qui renvoie `True` si l'entier représenté par `L` est pair, `False` sinon. *Cette fonction ne devra pas faire appel à la valeur de l'entier associé, mais utilisera uniquement la liste `L`.*



4.  Écrire une fonction `test(L:list, p:int) -> bool` avec `L` une liste représentant un entier naturel n codé sur N bits, p un entier avec $1 \leq p < N$ et qui renvoie `True` si $n < 2^p$ `False` sinon.



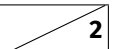
5.  On rappelle qu'en base 10, $311 > 255$ car $3 > 2$ et $311 > 307$ car les chiffres des centaines sont égaux et $1 > 0$. Compléter la fonction suivante afin qu'elle renvoie le plus grand des deux entiers représentés par les listes `A` et `B`.

```
def max_iter(A:list, B:list) -> bool:
    i, N = 0, len(A)
    while i < N:
        if A[i] > B[i]:
            return A
        elif A[i] < B[i]:
            _____
        else:
            i = _____
    _____
```

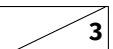
Exercice 2 Représentation des entiers signés

1.  Donner l'ensemble des entiers relatifs représentables sur 8 bits.

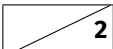


2.  2 Donner la représentation sur 8 bits de l'entier -57. On détaillera soigneusement la réponse.

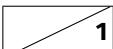


3.  3 Écrire une fonction `dec2binSigne(n:int, N:int) -> list` renvoyant la liste formant l'écriture binaire sur N bits de l'entier signé d'écriture décimale n sans nécessairement utiliser la technique du complément à deux. On supposera connue la fonction `dec2bin(m:int;N:int) -> list` renvoyant la liste formant l'écriture binaire sur N bits de l'entier naturel d'écriture décimale m).

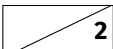


2.  2 Dans l'écriture ci-dessus, donner la mantisse M et justifier par le calcul que : $1 \leq M < 2$.



3.  1 Rappeler l'intervalle des exposants E représentables.



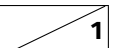
4.  2 Déterminer une approximation (en approchant la mantisse) de l'écart minimum entre deux réels d'exposant maximal représentables, et deux réels d'exposant minimal représentables.



Exercice 3 Représentation des réels On rappelle le principe d'écriture normalisée en base deux d'un réel non nul à virgules flottantes sur 64 bits :

$$(x)_2 = (-1)^S \times 2^E \times \left[1 + \sum_{i=1}^{52} \frac{b_i}{2^i} \right].$$

On notera e le nombre de bits dédiés au codage de l'exposant et m le nombre de bits utilisés pour coder la mantisse.

1.  1 Que vaut e ? Expliquer.



Solution 1

1. Ici $N = 8$, donc l'ensemble des entiers représentables est $\llbracket 0, 2^N - 1 \rrbracket = \llbracket 0, 255 \rrbracket$.
2. On trouve après calculs successifs des restes : $57 = (00111001)_2$.
3. Pour tester la parité, il suffit de regarder si le dernier bit (le plus petit) vaut 0. D'où la fonction ci-dessous.

```
def pair(L:list)->bool:
    for e in L:
        assert e == 0 or e == 1
    return L[-1] == 0
```

```
>>> L_57 = [0, 0, 1, 1, 1, 0, 0, 1]
>>> pair(L_57)
False
```

4. Pour effectuer le test, il suffit de regarder si : $\forall k \in \llbracket 0, N - p - 1 \rrbracket, L[k] = 0$.

```
def test(L:list, p:int)->bool:
    N = len(L)
    assert 1 <= p < N
    k = 0
    while k < N-p and L[k] == 0: # vérification de la \
        ↪ présence N-p zéros en début de liste
        k += 1
    return k == N-p
```

```
>>> L_57 = [0, 0, 1, 1, 1, 0, 0, 1]
>>> test(L_57, 5)
False
>>> test(L_57, 6)
True
```

5.

```
def max_iter(A:list, B:list)->list:
    i, N = 0, len(A)
    while i < N:
        if A[i] > B[i]:
            return A
        elif A[i] < B[i]:
            return B
        else:
            i += 1
```

```
return A # cas d'égalité
```

```
>>> L_53 = [0, 0, 1, 1, 0, 1, 0, 1]
>>> max_iter(L_53, L_57)
[0, 0, 1, 1, 1, 0, 0, 1]
```

Solution 2

1. Ici on a toujours $N = 8$. Les entiers signés représentables sont ceux de $\llbracket -2^{N-1}, 2^{N-1} - 1 \rrbracket = \llbracket -128, 127 \rrbracket$.
2. L'entier -57 est négatif, on code donc plutôt $-57 + 2^8 = 199$ sur 8 bits. On obtient l'écriture $(11000111)_2$.

3.

```
def dec2bin(n : int, N : int) -> list :
    assert 0 <= n < 2**N
    L = []
    while len(L) < N:
        L = [n%2]+L
        n = n//2
    return L
```

```
def dec2binSigne(n:int, N:int)->list:
    assert -2**(N-1) <= n < 2**(N-1)
    if n >= 0:
        return dec2bin(n, N)
    else :
        return dec2bin(n+2**N, N)
```

```
>>> dec2bin(199, 8)
[1, 1, 0, 0, 0, 1, 1, 1]
>>> dec2binSigne(-57, 8)
[1, 1, 0, 0, 0, 1, 1, 1]
>>> dec2bin(71, 8)
[0, 1, 0, 0, 0, 1, 1, 1]
```

Solution 3

1. On a la relation $1 + e + m = 64$ et ici $m = 52$ d'après la somme. Donc $e = 11$.

2. On a par définition $M = 1 + \sum_{i=1}^{52} \frac{b_i}{2^i}$. Or $b_i \leq 1$ pour tout $i \in \llbracket 1, 52 \rrbracket$. Donc :

$$1 \leq M = 1 + \sum_{i=1}^{52} \frac{b_i}{2^i} \leq \sum_{i=0}^{52} \left(\frac{1}{2^i} \right) = \frac{1 - 2^{-53}}{1 - \frac{1}{2}} < 2.$$

Donc : $\boxed{1 \leq M < 2}$.

3. $E \in \llbracket -2^{e-1} + 1; 2^{e-1} \rrbracket$. Donc $\boxed{\llbracket -2^{10} + 1; 2^{10} \rrbracket = \llbracket -1023, 1024 \rrbracket}$.

4. On s'intéresse à la différence entre deux nombres réels successifs x et x' représentés, d'exposant k fixé. La plus petite variation de mantisse possible est $\delta M = 2^{-m} = 2^{-52}$. Si on note $\delta x = |x' - x|$ avec $x = (-1)^s \times 2^k \times M$, on a $\delta x = 2^{k-m}$.

Ainsi :

- pour les plus petits nombres, $k = -1022$, d'où $\delta x = \boxed{2^{-1074}} \ll 1$.
- Pour les plus grands nombres, $k = 1023$, d'où $\delta x = \boxed{2^{971}} \gg 1$.