TP n° 02 – Représentation des nombres

I Dépassement de capacité

Question 1 : Écrire dans le tableau suivant le float (simple précision) le plus grand exprimable.

<table>
<tr><td>S</td><td colspan="8">Exposant</td><td colspan="23">Mantisse</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td><td colspan="4"></td></tr>
</table>

Question 2 : Calculer sa valeur dans la base décimale.

Question 3 : Calculer la valeur dans la base décimale du float (double précision) le plus grand exprimable.

Question 4 : Entrer dans une console python les commandes suivantes et en déduire si le type de float utilisé est de précision simple ou double.

```
>>> 2.0**(1023)
>>> 2.0**(1024)
>>> import sys
>>> sys.float_info.max
```

Question 5 : Calculer la valeur minimale (> 0) pour le type de variable utilisé par votre système.

Question 6 : Vérifier votre calcul grâce à la commande suivante.

```
>>> import sys
>>> sys.float_info.min
```

II Approximation de calcul

Lors de la première séance, nous avons remarqué que certains calculs étaient approximés. L'exemple suivant avait été utilisé.

```
>>> 1-1/3.-1/3.-1/3.
```

Question 7 : Calculer le nombre binaire permettant de définir le réel le plus proche de 1/3.

Question 8 : Écrire ce nombre sous la forme suivante $A * 2^{exp}$, où A est un entier, et exp, l'exposant le plus petit qui permet à A d'être un entier. Vous ferez ce calcul pour un float simple et un float double. (Le calcul sera plus simple si une similitude entre les deux calculs est trouvée)

Question 9 : Déterminer dans ces deux cas la valeur du décimal le plus proche de 1/3, dans le cas des deux types de float.

Question 10 : Calculer alors la valeur de l'opération $1 - 1/3. - 1/3. - 1/3.$ en prenant ce nombre approché et comparer cette valeur à celle trouvée en faisant le calcul directement dans la console python.

III Correction

III.1 Dépassement de capacité

Question 1 :

<table>
<tr><td>S</td><td colspan="8">Exposant</td><td colspan="23">Mantisse</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td colspan="4">7</td><td colspan="4">F</td><td colspan="4">7</td><td colspan="4">F</td><td colspan="4">F</td><td colspan="4">F</td><td colspan="4">F</td><td colspan="4">F</td></tr>
</table>

Question 2 :

Signe	Exposant	Mantisse
0	$\underbrace{111...110}_{8bits}$	$\underbrace{111...111}_{23bits}$

— Exposant : $2^8 - 2 = 254$, exposant simple $254 - 127 = 127$,
— Le chiffre à calculer est donc $\underbrace{111...111}_{24bits}\underbrace{000...000}_{104bits}$,
— Ce qui donne en décimal $(2^{24} - 1) * 2^{104} = 3.4028234663852886.10^{38}$

Question 3 :

Signe	Exposant	Mantisse
0	$\underbrace{111...110}_{11bits}$	$\underbrace{111...111}_{52bits}$

— Exposant : $2^{11} - 2 = 2046$, exposant simple $2046 - 1023 = 1023$,
— Le chiffre à calculer est donc $\underbrace{111...111}_{53bits}\underbrace{000...000}_{971bits}$,
— Ce qui donne en décimal $(2^{53} - 1) * 2^{971} = 1.7976931348623157.10^{308}$

Question 5 :

Signe	Exposant	Mantisse
0	$\underbrace{000...001}_{11bits}$	$\underbrace{000...000}_{52bits}$

— Exposant : 1, exposant simple $1 - 1023 = -1022$,
— Le chiffre à calculer est donc $1,\underbrace{000...000}_{-1022bits}$,
— Ce qui donne en décimal $1 * 2^{-1022} = 2.2250738585072014.10^{-308}$

Question 7 :

$$\begin{array}{lllllllll}
0{,}33.. & \text{x} & 2 & = & 0{,}66.. & = & 0 & + & 0{,}66.. \\
0{,}66.. & \text{x} & 2 & = & 1{,}33.. & = & 1 & + & 0{,}33.. \\
0{,}33.. & \text{x} & 2 & = & 0{,}66.. & = & 0 & + & 0{,}66.. \\
0{,}66.. & \text{x} & 2 & = & 1{,}33.. & = & 1 & + & 0{,}33.. \\
0{,}33.. & \text{x} & 2 & = & 0{,}66.. & = & 0 & + & 0{,}66.. \\
\ldots
\end{array}$$

On remarque un récurrence dans l'écriture du $0,33_{10}$ en binaire : $0,33.._{10} = 0,01010.._2$

Question 8 :
— 32 bits : $1,\underbrace{0101..}_{23bits} * 2^{-2} = \underbrace{10101010....}_{24bits} * 2^{-25}$
— 64 bits :$1,\underbrace{0101..}_{52bits} * 2^{-2} = \underbrace{10101010....}_{53bits} * 2^{-54}$

Le passage de $\underbrace{10101010...0}_{24bits}$ à $\underbrace{10101010...1}_{23bits}$ se fait par un décalage des bits vers la droite ce qui revient à diviser par deux.
Le passage de $\underbrace{10101010...1}_{53bits}$ à $\underbrace{10101010...0}_{54bits}$ se fait par un décalage des bits vers la gauche ce qui revient à multiplier par deux.

Question 9 :

- 32 bits : $\underbrace{10101010...0}_{24bits} = \underbrace{11111111....}_{24bits} - \underbrace{10101010...1}_{23bits}$
- 64 bits : $\underbrace{10101010...1}_{53bits} = \underbrace{11111111....}_{54bits} - \underbrace{10101010...0}_{54bits}$

Le calcul de $\underbrace{11111111....}_{24bits}$ se fait en ajoutant 1.

- 32 bits : $\underbrace{10101010...0}_{24bits} = (2^{24} - 1) - \frac{\overbrace{10101010...0}^{24bits}}{2}$
- 64 bits : $\underbrace{10101010...1}_{53bits} = (2^{54} - 1) - 2 * \underbrace{10101010...1}_{53bits}$

Regroupement des $\underbrace{10101010...0}_{24bits}$ et $\underbrace{10101010...1}_{53bits}$.

- 32 bits : $\underbrace{10101010...0}_{24bits} = \frac{2}{3}.(2^{24} - 1)$
- 64 bits : $\underbrace{10101010...1}_{53bits} = \frac{1}{3}.(2^{54} - 1)$

Le résultat est alors.

- 32 bits : $\underbrace{10101010...0}_{24bits} * 2^{-25} = \frac{2}{3}.(2^{24} - 1) * 2^{-25} =$
- 64 bits : $\underbrace{10101010...1}_{53bits} * 2^{-54} = \frac{1}{3}.(2^{54} - 1) * 2^{-54} =$

Question 10 :

- 32 bits : $1 - 1/3. - 1/3. - 1/3. = 1 - 2.(2^{24} - 1) * 2^{-25} = 1 - (2^{24} - 1) * 2^{-24} = 2^{-24} = 5.960464477539063.10^{-8}$
- 64 bits : $1 - 1/3. - 1/3. - 1/3. = 1 - 1.(2^{54} - 1) * 2^{-54} = 1 - (2^{54} - 1) * 2^{-54} = 2^{-54} = 5,55111512312578.10^{-17}$