TD 3

Exercice 1. Représentation des nombres à virgule flottante

1 Donner l'écriture des nombres 1, 2, 3, 4 et 5 au format IEEE 754 en simple précision.

2 Si l'exposant biaisé est différent de 0 et 255 (ou 2047 en double précision), quels sont les plus petits et les plus grands nombres strictement positifs représentables en simple et double précision ?

3 Même question avec un exposant biaisé valant 0 (représentation dé-normalisée). Quel est le plus petit
nombre strictement positif représentable ?

Exercice . Représentation des nombres à virgule flottante

1 Donner l'écriture des nombres 1, 2, 3, 4 et 5 au format IEEE 754 en simple précision.

1 = + 1,0 x 2^{0} d'où signe = 0 , mantisse = 000….000 ,
exposant = +127 10 = 01111111
soit : **0 01111111 00000000000000000000000**

2 = 10 = 1,0 x 2^{1} soit : **0 10000000 00000000000000000000000**

3 = 11 = 1,1 x 2^{1} soit : **0 10000000 10000000000000000000000**

4 = 100 = 1,0 x 2^{2} soit : **0 10000001 00000000000000000000000**

Exercice . Représentation des nombres à virgule flottante

Si l'exposant biaisé est différent de 0 et 255 soit 1 à 254 (ou 2047 en double précision), quels sont les plus petits et les plus grands nombres strictement positifs représentables en simple et double précision ?

Simple	$1.0\ 2^{-126}$ / $1{,}17\ 10^{-38}$	$1{,}1111\ 2^{127}$ / $3{,}4\ 10^{38}$
Double	$1.0\ 2^{-1022}$ / $2{,}22\ 10^{-308}$	$1{,}111\ 2^{1023}$ / $1{,}79\ 10^{308}$

Exemple de valeurs *(source Wikipédia)*

Age de l'univers : $13{,}8\ 10^{9}$ ans = 13,8 milliards d'années

Taille de l'univers : $8{,}8\ 10^{23}$ Km = 880 000 milliards de milliards de Km

Poids du soleil : $2\ 10^{30}$ Kg

Volume de l'univers : 10^{81} M^3

Taille d'un atome : 10^{-10} à 10^{-15} m

Poids d'un atome : 10^{-25} à 10^{-27} km

Mais encore …..

341 10^6 ordinateurs vendus en 2022 dans le monde
6,2 10^9 ordinateurs en service dans le monde

3,1 10^6 ordinateurs vendus en France en 2021
2,5 10^6 ordinateurs mis au rebus en France en 2021

A méditer ….

Exercice . Représentation des nombres à virgule flottante
Même question avec un exposant biaisé valant 0 (représentation dénormalisée). Quel est le plus petit nombre strictement positif représentable ?

avec exp biaisé = 0

	Mini	Maxi
Simple	$0{,}000\ \ 2^{-127}$ / $1{,}4\ 10^{-45}$	$0{,}111\ 2^{-127}$ / $1{,}17\ 10^{-45}$
Double	$0{,}000\ 2^{-1023}$ / $5{,}0\ 10^{-324}$	$0{,}111\ 2^{-1023}$ / $2{,}22\ 10^{-324}$

Donner les représentations en simple précision à virgule flottante de 25 et 2,125.

2^5

2^5 = 100000 = 1,0 x 2^5 → signe = 0 exposant = 127+5=132 = 10000100 Mantisse = 000000000

2^5 = 0 10000100 00000000000000000000000

2,125

$2_{10} = 10_2$

$0{,}125_{10}$

0,125 * 2 = 0,250 → 0
0,250 * 2 = 0,500 → 0
0,500 * 2 = 1,0 → 1
0 * 2 = 0 → 0

d'où $0{,}125_{10} = 0{,}001_2$ et $2{,}125_{10} = 10{,}001_2$

2,125 = 10,001 x 2^0 = 1,0001 x 2^1 soit signe =0 , mantisse = 00010… Exposant = 128 = 10000000

= 0 10000000 00010000000000000000000

Quelle est la représentation en simple précision à virgule flottante de 1/10 ? Quelle est l'erreur obtenue ?

0,1 = 0,0001100110011001100110011….x 2^0
= 1,1001100110011001100110011. ..x 2^{-4}
Soit s = 0 , Exp = 127+(-4) = 123

0,1 = **0 01111011 10011001100110011001100** en IEEE 754

La régénération donne 0,099998 soit presque 0,1

Même question pour 1/5

1/5 = 0,2 = 0,1x2 d'où 1/5 = 0,00110011001100110011…

1/5 = 0 01111100 10011001100110011001100

La régénération donne 0,199998 soit presque 0,2

Perte d'information en arithmétique flottante
(Vous pouvez écrire de petits programmes sur machine pour vous aider)

Soient les nombres
X = 0 10001110 00000000000000000000000
Y = 0 10011010 00000000000000000000000
au format IEEE 754 en simple précision.

1 Donnez les représentations des nombres
a. A = X + 1
b. B = X + Y
c. C = Y + A

Perte d’information en arithmétique flottante

Perte d’information en arithmétique flottante

A = 0 10001110 00000000000000000000000
B = 0 10011010 00000000000000000000000

au format IEEE 754 en simple précision.

Soit $A = 2^{142-127} \times 1,0 = 2^{15}$

$B = 2^{154-127} \times 1,0 = 2^{27}$

(a) C = A + 1

A = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

C = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 x 2^0
= 1,0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 x 2^{15}

(b) D = A + B

A = 1,0 x 2^{15} = 0 , 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1x2^{27}
B = 1,0 x 2^{27} = 1 , 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0x2^{27}

D = 1, 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1x 2^{27}

(c) E = B + C

C = 1,0000000000000001 x 2^{15}
B = 1,0 x 2^{27}

C = 0,00000 00000 00100 00000 00000 001 x 2^{27}
B = 1,00000 00000 00000 00000 00000 000 x 2^{27}

E = 1,00000 00000 00100 00000 00000 001 x 2^{27}

$E = 1{,}00000\ 00000\ 00100\ 00000\ 00000\ 001 \times 2^{27}$

$1{,}00000\ 00000\ 00100\ 00000\ 000\mathbf{\sout{00\ 001}} \times 2^{27}$

Pour le codage en IEEE 754, on ne garde que 23 bits pour la mantisse , on va donc perdre les 5 derniers bits, soit 00001

D'où : C+B → B, la valeur de B très grande absorbe C.

Un premier programme

```
A = 2.000000
A = 4.000000
A = 8.000000
A = 16.000000
A = 32.000000
A = 64.000000
A = 128.000000
A = 256.000000
A = 512.000000
A = 1024.000000
A = 2048.000000
A = 4096.000000
…
A = 57646075230342349O.000000
A = 115292150460684700O.000000
A = 230584300921369400O.000000
A = 461168601842738790O.000000
A = 922337203685477580O.000000
A = 184467440737095520O.000000
B = 2.000000
```

```
A=1;
while ((((A + 1) - A) - 1) == 0)
{ A=2*A;
  printf ("A = %f\n",A);
}
B=1;
while  ((((A + B) - A) - B) != 0)
{ B=B+1;
  printf ("B = %f\n",B);
}
```

RAPPELS : float a,b;

1,0000 0000 0000 0000 0000 0000 2^{23} A

\+ 0,0000 0000 0000 0000 0000 0001 2^{23} 1

1,0000 0000 0000 0000 0000 0001 2^{23} A

d'où *((((A + 1) - A) - 1) = -1*

\+ 0,0000 0000 0000 0000 0000 0010 2

1,0000 0000 0000 0000 0000 0000 2^{23} A

1,0000 0000 0000 0000 0000 0011 A+2

d'où *((((A + B) - A) - b) = 0*

Un deuxième programme : sommes

```
n= 10 000
S1 = 9.787613 , S2=9.787604

n = 100 000
S1 = 12.090851 , S2=12.090152

n = 1 000 000
S1 = 14.357358 , S2=14.392652

n = 5 000 000
S1 = 15.403683 , S2=16.007854

n = 6 000 000
S1 = 15.403683 , S2=16.182493
```

$$S1_n = \sum_{i=1}^{n} 1 / i \qquad S2_n = \sum_{i=n}^{1} 1 / i$$

RAPPELS : float s1,s2;