1ère Spé. NSI Gr. 2 **Contrôle #2, 18/12/2023**
Lycée Fustel de Coulanges **Durée: 1 Heure**

Ce contrôle comporte 7 questions; il sera noté sur 20 points. Les réponses sont à porter sur une copie comportant votre nom. Il n'est pas nécessaire de répondre aux questions dans l'ordre — commencez par celles où vous vous sentez le plus à l'aise *(mais ne tentez les questions bonus qu'après avoir fini le reste!!)*. Les calculatrices ne sont pas autorisées.

---

1. (1 ½ points) *Questions à choix multiples* (aucune justification de la réponse n'est nécessaire; plusieurs réponses sont possibles):
   (a) L'ordre correct, de la plus petite quantité à la plus grande, est:
   - ◯ **A:** 1 Mo (mégaoctet), 1 bit, 1 Go (gigaoctet), 1 To (téraoctet), 1 ko (kilooctet)
   - ◯ **B:** 1 bit, 1 ko, 1 Mo, 1 Go, 1 To
   - ◯ **C:** 1 bit, 1 ko, 1 Go, 1 To, 1 Mo
   - ◯ **D:** 1 bit, 1 ko, 1 Mo, 1 To, 1 Go

   (b) Sélectionnez, parmi les affirmations ci-dessous, celle(s) qui est (ou sont) vraie(s):
   - ◯ **A:** Un nombre réel encodé au moyen de la norme IEEE 754 ne peut jamais être une valeur exacte – c'est toujours une valeur approchée.
   - ◯ **B:** Il est dangereux de faire des comparaisons entre nombres flottants car du fait de leur encodage cela peut générer des erreurs.
   - ◯ **C:** L'encodage des nombres réels au moyen de la norme IEEE 754 s'appuie sur les mêmes principes que la notation scientifique (sauf que l'on utilise des puissances de 2 au lieu de puissances de 10).

   (c) Sélectionnez, parmi les affirmations ci-dessous, celle(s) qui est (ou sont) vraie(s):
   - ◯ **A:** Un texte codé en ASCII est simplement une suite d'octets correspondant au codage de chacun des caractères du texte.
   - ◯ **B:** Un texte codé en ASCII ne peut pas contenir de lettres comportant des accents (comme "é" par exemple).
   - ◯ **C:** L'encodage ASCII est moins complet que l'encodage UTF-8.
   - ◯ **D:** Un caractère codé en ASCII l'est sur 64 bits.

2. *Addition de nombres binaires* (posez bien votre addition – un simple résultat ne sera pas accepté.)
   (a) (1 point) Effectuez l'addition suivante: 10001011 + 10001110
   (b) (1 point) Est-ce-que cette addition peut être effectuée sans erreur si les entiers sont codés sur 8 bits? Justifiez votre réponse.

3. *Conversion entre bases de numération* (le détail des calculs est demandé – le résultat seul ne rapportera pas la totalité des points.)
   (a) (1 point) Convertissez de base 10 en base 2: $26_{(10)}$
   (b) (1 point) Convertissez de base 16 en base 2: $2B_{(16)}$

4. *Entiers relatifs*

   On considère le nombre binaire $10011100_{(2)}$.
   (a) (1 point) Quelle est la valeur en base 10 de ce nombre, s'il représente un entier non signé sur un octet[1]?
   (b) (1 point) Quel est le complément à un de 10011100?

[1] Au cas où ça pourrait vous être utile: $2^7 = 128$

(c) (1 point) Quel est le complément à deux de 10011100?

(d) (1 point) Quelle est la valeur en base 10 du nombre binaire 10011100 s'il représente un entier signé?

5. *Chiffrement*

On considère le code python ci-dessous[2].

```
message = ["M", "A", "X", "!"]
decal = 5
resultat = ""

# Processus de transformation
for i in range(len(message)):
    caractere = message[i]
    if ord("A") <= ord(caractere) <= ord("Z"):
        code_ascii = ord(caractere)
        print(code_ascii) # PRINT #1
        nouveau_code = code_ascii + decal
        nouveau_caractere = chr(nouveau_code)
        print(nouveau_caractere) # PRINT #2
        resultat = resultat + nouveau_caractere
    else:
        resultat = resultat + caractere

# Affichage final
print ("Resultat :", resultat ) # PRINT #3
```

(a) (2 points) Quel est l'affichage en console si l'on exécute ce code? (notez qu'il y a 3 print dans l'ensemble du code.) A quoi pourrait servir ce code?

On pourra s'aider de l'extrait de la table ascii ci-dessous :

65 : A	66 : B	67 : C	68 : D	69 : E	70 : F	71 : G	72 : H	73 : I	74 : J
75 : K	76 : L	77 : M	78 : N	79 : O	80 : P	81 : Q	82 : R	83 : S	84 : T
85 : U	86 : V	87 : W	88 : X	89 : Y	90 : Z	91 : [	92 : \	93 :]	94 : ^
95 : _	96 : ‘	97 : a	98 : b	99 : c	100 : d				

(b) (1 point) Quel est le problème avec la lettre "X"?

(c) (1 point) Comment pourriez-vous résoudre ce problème?

6. *Parcours d'une liste*

On vous demande de coder une fonction CodeASCII(lst) qui prend en entrée une liste de caractères et renvoie en sortie une liste de même longueur contenant le code ASCII de chacun des caractères. Par exemple (en se référant à l'extrait de la table ASCII de l'exercice précédent) on aurait `CodeASCII(["M", "A", "X"]) = [77, 65, 88]`. *Note: il est évident que l'on se servira ici de la fonction ord() utilisée dans l'exercice précédent*[2].

(a) (2 points) Rédigez dans un premier temps l'algorithme qui sera mis en œuvre par une telle fonction – idéalement sous forme de pseudo-code.

(b) (2 points) Rédigez dans un second temps le code python de la fonction CodeASCII(lst).

7. *Codage des flottants – norme IEEE 754 à simple précision*

L'objectif de cet exercice est de découvrir quel est le nombre réel codé par 11000000110100000000000000000000.

Ce nombre sera appelé N dans cet exercice.

On rappelle que le premier bit est `?????????????` (noté S), les 8 bits suivants correspondent à `?????????????` (noté E), et les 23 bits suivants à `?????????????` (noté M).

Ainsi, on a $S = 1$, $E = 10000001$ et $M = 10100000000000000000000$.

On rappelle que $2^{-1} = 0.5$, $2^{-2} = 0.25$, $2^{-3} = 0.125$.

On rappelle la formule suivante : $N = (-1)^S \times (1 + M) \times 2^{E-127}$.

[2]Deux rappels: `ord("X")` renvoie le code ASCII du caractère "X"; `chr(88)` renvoie le caractère dont le code ASCII est 120.

(a) (1 point) Écrivez sur votre feuille, dans le bon ordre, les mots qui doivent figurer à la place des trois "`?????????????`" dans l'énoncé de cet exercice.

(b) (½ point) N est-il positif ou négatif?

(c) (½ point) Quelle est la valeur de E?

(d) (½ point) Quelle est la valeur de M?

(Question bonus 1)

Code mystère: quel est l'affichage obtenu en console si on exécute ce code? Que signifie-t-il / que fait ce code? On ne demande pas de détailler les étapes du calcul, mais d'expliciter le lien entre les variables de départ et ce qui est affiché à la fin du programme. (conseil: commencez par exécuter ce programme "à la main" pour voir ce qui se passe à chaque étape)

```
x = 97
puissance_2 = 2**7 #(ce qui vaut 128)
res = ""
while puissance_2 >= 1:
    if puissance_2 <= x:
        res += "1"
        x = x - puissance_2
    else:
        res += "0"
    puissance_2 = puissance_2 / 2
print (res)
```

(Question bonus 2)

En cours on a codé une fonction CompUn(lst) qui renvoie le complément à un d'une liste de bits représentant un entier codé en binaire. On vous demande de coder une deuxième fonction, qu'on appellera Ajouter1(lst), qui prendra le résultat de la précédente, ajoutera un, et renverra donc le complément à deux de l'entier initial.

Quelques remarques:

- La syntaxe pour une boucle bornée dont l'indice va descendre de N à 0 est `for i in range(N,-1,-1):`
- On ignorera le cas d'une liste uniquement composée de 1 (et pour laquelle un ajout de un ajouterait un chiffre).
- Conseil: commencez par faire à la main `100111 + 1` et réfléchissez aux étapes que vous accomplissez, à comment vous gérez les retenues, à ce qui se passe quand il n'y a plus de retenue...