CORRIGÉ DÉTAILLÉ • ÉPREUVE 1

1ère NSI • Numération, Plages de valeurs, Complément à 2 & Débordement • Barème officiel / 20

CORRIGÉ OFFICIEL • ÉLÈVES

Partie 1 : QCM Défi Numération (Règle stricte +1 / -1 / 0) — 4 points

Barème QCM : Bonne réponse = +1 pt | Mauvaise réponse = -1 pt | Absence = 0 pt • Sous-total de section ≥ 0 pt

Q1 : Particularité de -128 (1000 0000_2) • **Réponse B** *(1 pt)*
Inversion (0111 1111_2) + 1 redonne 1000 0000_2. L'opposé (+128) ne peut pas être représenté sur 8 bits signés (max +127), provoquant un débordement.

Q2 : Addition binaire X + ~X • **Réponse B** *(1 pt)*
Toujours **1111 1111_2** (255 ou -1). Pour chaque rang binaire *i*, soit (0+1) soit (1+0) = 1 sans retenue.

Q3 : Bits minimum pour 1 000 000 • **Réponse C** *(1 pt)*
2^{19} = 524 288 < 1 000 000 ≤ 2^{20} = 1 048 576 ⇒ **20 bits minimum** sont obligatoires.

Q4 : Addition (+112) + (+48) • **Réponse B** *(1 pt)*
+112 + +48 = +160. Comme +160 > +127, le bit de signe devient 1 (1010 0000_2 = -96). Il y a un **débordement (overflow)**.

Partie 2 : Plages de valeurs & Entiers signés en Complément à 2 — 6 points

Exercice 2.1 : Plages de valeurs & Codage de -43 — 3,5 points

CORRIGÉ EX 2.1 • MÉTHODE PAS À PAS

1. a) Non signé : 2^8 = 256 valeurs ⇒ **[0 ; 255]** *(0,5 pt)*.
b) Signé (complément à 2) : **[-128 ; +127]** *(0,5 pt)*.
c) **Dissymétrie :** Le zéro (0000 0000_2) ayant son bit de signe à 0, il est comptabilisé avec les positifs. Il y a donc 127 entiers strictement positifs (+1 à +127) et 128 entiers négatifs (-1 à -128). *(1 pt)*

2. Encodage de -43 :
• Étape 1 : Valeur absolue (+43) = 32 + 8 + 2 + 1 = **$0010\ 1011_2$** *(0,5 pt)*
• Étape 2 : Complément à 1 (inversion bit à bit) = **$1101\ 0100_2$** *(0,5 pt)*
• Étape 3 : Complément à 2 (+ 1) = 1101 0100_2 + 1 = **$1101\ 0101_2$** *(0,5 pt)*
Vérification : -128 + 64 + 16 + 4 + 1 = -128 + 85 = -43.

Exercice 2.2 : Arithmétique binaire & Débordement (Overflow) — 2,5 points

CORRIGÉ EX 2.2 • ANALYSE DE L'OVERFLOW

1. Addition posée :
Retenues : **1 1 0 0 0 1 1 0**
0 1 1 0 1 0 0 1_2 (+105)
\+ 0 0 1 1 0 0 1 1_2 (+51)
= **$1\ 0\ 0\ 1\ 1\ 1\ 0\ 0_2$** *(1,5 pt)*

2. Interprétation : Le bit de poids fort vaut **1**, ce qui correspond en complément à 2 à un nombre **négatif (-100)** alors qu'on additionnait deux nombres positifs (+105 + +51 = +156). Comme +156 dépasse le maximum représentable (+127), on a un **débordement de capacité (overflow)**. *(1 pt)*

Partie 3 : Conversions de l'année 2026 & Algorithme Python — 10 points

Exercice 3.1 : Décomposition & Divisions successives de 2026 — 5 points

CORRIGÉ EX 3.1 • DEUX MÉTHODES DE CONVERSION

1. $2^{10} = 1024 \leq 2026 < 2048 = 2^{11}$ ⇒ Il faut exactement **11 bits**. *(1 pt)*

2. Méthode des puissances :

$2026 = 1024 + 512 + 256 + 128 + 64 + 32 + 8 + 2 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^3 + 2^1$.

⇒ **2026_{10} = $(111\ 1110\ 1010)_2$**. *(2 pts)*

3. Divisions successives par 2 :

2026/2=1013 (r=0) • 1013/2=506 (r=1) • 506/2=253 (r=0) • 253/2=126 (r=1) • 126/2=63 (r=0) • 63/2=31 (r=1) • 31/2=15 (r=1) • 15/2=7 (r=1) • 7/2=3 (r=1) • 3/2=1 (r=1) • 1/2=0 (r=1).

Lecture de bas en haut (dernier reste = bit de poids fort) : **$(111\ 1110\ 1010)_2$**. *(2 pts)*

Exercice 3.2 : Analyse d'un algorithme de conversion en Python — 5 points

CORRIGÉ EX 3.2 • TRACÉ D'EXÉCUTION & ANALYSE

1. Tableau d'état pour n = 21 : *(2 pts)*

- Tour 1 : n=21, reste = 21%2 = 1, binaire = "1", n=10
- Tour 2 : n=10, reste = 10%2 = 0, binaire = "01", n=5
- Tour 3 : n=5, reste = 5%2 = 1, binaire = "101", n=2
- Tour 4 : n=2, reste = 2%2 = 0, binaire = "0101", n=1
- Tour 5 : n=1, reste = 1%2 = 1, binaire = "10101", n=0 ⇒ Arrêt et renvoi de "10101" (21_{10}).

2. Concaténation à gauche : Chaque reste successif correspond au bit du rang supérieur suivant. On doit placer le nouveau reste **à gauche** pour respecter le sens de lecture binaire usuel (bit de poids fort à gauche). *(1,5 pt)*

3. Itérations pour n = 2026 : 2026 s'écrit sur 11 bits. Chaque tour de boucle réalise une division entière par 2 (`n // 2`), la boucle s'exécutera donc exactement **11 fois**. *(1,5 pt)*

PARTIE	INTITULÉ DU THÈME ÉVALUÉ	BARÈME	NOTE MAX
Partie 1	QCM Défi Numération (Règle stricte +1 / -1 / 0)	4 pts	/ 4
Partie 2	Plages de valeurs, Complément à 2 & Débordement	6 pts	/ 6
Partie 3	Conversions de l'année 2026 & Algorithme Python	10 pts	/ 10
	TOTAL ÉPREUVE 1 :	**20 pts**	**/ 20**

CORRIGÉ DÉTAILLÉ • ÉPREUVE 2

1ère NSI • Hexadécimal, Table ASCII avancée & Bit-Packing d'une Date • Barème officiel / 20

CORRIGÉ OFFICIEL • ÉLÈVES

Partie 1 : QCM Défi Hexadécimal & Manipulation de Bits (Règle stricte +1 / -1 / 0) — 4 points

Barème QCM : Bonne réponse = +1 pt | Mauvaise réponse = -1 pt | Absence = 0 pt • Sous-total de section ≥ 0 pt

Q1 : Chiffres hexa pour $2^{33} - 1$ • **Réponse B** *(1 pt)*
2^{33} - 1 s'écrit sur 33 bits à 1. Comme 1 chiffre hexa = 4 bits : 8 chiffres couvrent 32 bits (< 33), il faut donc **9 chiffres** (couvrant jusqu'à 36 bits).

Q2 : 'C' avec bit 5 forcé à 1 • **Réponse A** *(1 pt)*
Le bit 5 a pour poids 2^5 = 32. Forcer ce bit à 1 ajoute 32 au code ASCII : 'C' (67) + 32 = 99 = code de **'c' (minuscule)**.

Q3 : Python (0b11101 >> 2) • **Réponse B** *(1 pt)*
Un décalage à droite de 2 rangs supprime les 2 bits de droite : 0b11101 (29) >> 2 donne **0b111 = 7** (division entière 29 // 4).

Q4 : Extraire le vert (8A) de 0x3A8AFC • **Réponse A** *(1 pt)*
Le vert est aux bits 8-15. On décale de 8 bits vers la droite (`>> 8`), puis on applique le masque `& 0xFF` pour ne garder que l'octet bas.

Partie 2 : Hexadécimal & Encodage ASCII fin — 6 points

Exercice 2.1 : L'Hexadécimal & les quartets — 3 points

CORRIGÉ EX 2.1 • QUARTETS & BASE 16

1. Intérêt de la base 16 : L'écriture est 4 fois plus courte que le binaire (très lisible pour l'humain). La passerelle binaire/hexa est directe car 16 = 2^4 : 1 symbole hexa correspond exactement à 1 quartet (4 bits). *(0,5 pt)*

2. Découpage par quartets de A :

$(1011)_2$ = 11 = **B** • $(0000)_2$ = **0** • $(1111)_2$ = 15 = **F** • $(1101)_2$ = 13 = **D** ⇒ A = $\mathbf{(B0FD)_{16}}$ *(1,5 pt)*

3. Conversion de $(2F8)_{16}$:

$(2F8)_{16} = 2\times16^2 + 15\times16^1 + 8\times16^0 = 2\times256 + 240 + 8 = 512 + 248 = \mathbf{760_{10}}$. *(1 pt)*

Exercice 2.2 : Table ASCII & Propriété fine du bit de casse — 3 points

CORRIGÉ EX 2.2 • ASCII & BIT DE CASSE

1. Encodage de « NSI » :

• 'N' : 78 = 4×16 + 14 = $\mathbf{(4E)_{16}}$ = $\mathbf{0100\ 1110_2}$ • 'S' : 83 = 5×16 + 3 = $\mathbf{(53)_{16}}$ = $\mathbf{0101\ 0011_2}$ • 'I' : 73 = 4×16 + 9 = $\mathbf{(49)_{16}}$ = $\mathbf{0100\ 1001_2}$.

Taille mémoire : 3 caractères ASCII = **3 octets = 24 bits**. *(1,5 pt)*

2. Casse (Majuscule / Minuscule) :

• 'A' = $0100\ 0001_2$ (65) et 'a' = $0110\ 0001_2$ (97) ⇒ seul le **bit de rang 5** (poids 2^5 = 32) passe de 0 à 1. *(0,5 pt)*

• Déduction pour 'n' : On prend 'N' ($0100\ 1110_2$) et on passe le bit 5 à 1 ⇒ $\mathbf{0110\ 1110_2 = (6E)_{16}}$ (soit 78 + 32 = 110). *(1 pt)*

Partie 3 : Emballage binaire (Bit-Packing) d'une Date & Python — 10 points

Exercice 3.1 : Problème d'ingénierie — Stockage compact d'une date (24 bits) — 7 points

CORRIGÉ EX 3.1 • BIT-PACKING & DÉCOUPAGE PAR QUARTETS

1. Justifications par puissances de 2 : *(1,5 pt)*
• Jour J ≤ 31 : $2^4 = 16 < 31 \leq 2^5 = 32 \Rightarrow$ **5 bits** suffisent.
• Mois M ≤ 12 : $2^3 = 8 < 12 \leq 2^4 = 16 \Rightarrow$ **4 bits** suffisent.
• Année A ≤ 2047 : $2^{10} = 1024 < 2048 = 2^{11} \Rightarrow$ **11 bits** suffisent (plage [0; 2047]).

2. Conversions individuelles (28/09/2026) : *(2 pts)*
• J = 28 = 16 + 8 + 4 = $\mathbf{(11100)_2}$ (5 bits)
• M = 9 = 8 + 1 = $\mathbf{(1001)_2}$ (4 bits)
• A = 2026 = $\mathbf{(111\ 1110\ 1010)_2}$ (11 bits)

3. Mot binaire complet de 24 bits :
Par champs : [0000] [111 1110 1010] [1001] [11100] ⇒ Par quartets : $\mathbf{0000\ 1111\ 1101\ 0101\ 0011\ 1100_2}$ *(1,5 pt)*

4. Écriture hexadécimale :
Quartets : 0000=0, 1111=F, 1101=D, 0101=5, 0011=3, 1100=C ⇒ **0x0FD53C**. *(1 pt)*

5. Formule arithmétique Python :
J est décalé de 0 bit (×1), M est décalé de 5 bits (×2^5 = 32), A est décalé de 5+4 = 9 bits (×2^9 = 512).
⇒ **mot = A × 512 + M × 32 + J** (ou `(A << 9) | (M << 5) | J`). *Vérif : 2026×512 + 9×32 + 28 = 1 037 628 = 0x0FD53C. (1 pt)*

Exercice 3.2 : Questions de synthèse & Python (Site ndlrnsi) — 3 points

CORRIGÉ EX 3.2 • SYNTHÈSE PYTHON & WEB

1. `int("1101", 2)` vaut 13 et `int("A", 16)` vaut 10 ⇒ x = 13 + 10 = **23**. *(1 pt)*

2. `a = 0b1011` définit un entier décimal : Type = **int**, Valeur = **11**. `b = bin(11)` génère la représentation binaire textuelle : Type = **str**, Valeur = **'0b1011'**. *(1 pt)*

3. Dans `#1E3A8A`, la composante rouge est $(1E)_{16}$. $(1E)_{16} = 1 \times 16 + 14 = \mathbf{30_{10}}$ (soit une intensité de 30 / 255). *(1 pt)*

PARTIE	INTITULÉ DU THÈME ÉVALUÉ	BARÈME	NOTE MAX
Partie 1	QCM Défi Hexadécimal & Manipulation de Bits (Règle stricte +1 / -1 / 0)	4 pts	/ 4
Partie 2	Hexadécimal & Encodage ASCII fin	6 pts	/ 6
Partie 3	Bit-Packing d'une Date & Python	10 pts	/ 10
	TOTAL ÉPREUVE 2 :	**20 pts**	**/ 20**