

NOM :
Prénom :
Classe :

Q.C.M 5

55 minutes

1. Soit l'entier signé codé sur 8 bits $n = 1001\ 0110$. Son complément à deux est :
 - ☐ 0110 1001
 - ☒ **0110 1010**
 - ☐ 1110 1010
 - ☐ Cet entier n'a pas de complément à deux.
2. Pour effectuer la soustraction $a - b$ de deux entiers signés codés sur 8 bits, on procède ainsi :
 - ☐ Additionner b et le complément à 2 de a .
 - ☒ **Additionner a et le complément à 2 de b .**
 - ☐ Additionner a et b et prendre le complément à 2 du résultat.
 - ☐ Prendre le NOT du bit de poids fort de b et additionner ce résultat avec a .
3. On considère le demi additionneur de deux bits réalisant l'addition $A + B$.
 - ☐ La somme $S = A \text{ and } B$; La retenue $C = A \text{ or } B$.
 - ☐ La somme $S = A \text{ or } B$; La retenue $C = A \text{ and } B$.
 - ☐ La somme $S = A \text{ and } B$; La retenue $C = A \text{ xor } B$.
 - ☒ **La somme $S = A \text{ xor } B$; La retenue $C = A \text{ and } B$.**
4. Voici deux boucles :
Boucle 1 :

```
for i in range(50000) :  
    a = 1,001**i
```


Boucle 2 :

```
for i in range(50000) :  
    a = 2**i
```

 - ☒ **La boucle 1 s'exécute plus rapidement que la boucle 2.**
 - ☐ La boucle 2 s'exécute plus rapidement que la boucle 1.
 - ☐ Les deux ont à peu près la même durée.
 - ☐ On ne peut pas savoir à l'avance.
5. Voici une copie d'un terminal Linux dans lequel on a saisi et exécuté plusieurs commandes :

```
m1@rasp_m1:~ $ pwd  
/home/m1  
m1@rasp_m1:~ $ ls  
Bureau  Documents  Images  Modèles  Musique  Public  Téléchargements  Vidéos  
m1@rasp_m1:~ $ cd Documents  
m1@rasp_m1:~/Documents $ mkdir NSI  
m1@rasp_m1:~/Documents $ ls
```

Quel peut être l'affichage après l'exécution de cette dernière commande ?

 - ☒ **mu_raspberry spyder_raspberry jupyter_notebook_raspberry NSI**
 - ☐ Bureau Documents Images Modèles Musique Public Téléchargements Vidéos
 - ☐ Documents ls
 - ☐ mu_raspberry spyder_raspberry jupyter_notebook_raspberry

6. Si on travaille avec des entiers signés codés sur 8 bits, alors l'addition $64 + 66$ affiche
- ☐ -130
 - ☐ -129
 - ☒ -126
 - ☐ 130
7. L'expression $0,1 + 0,2 > 0,3$ a la valeur True. Quelle en est la raison ?
- ☐ C'est parce que la machine utilise des mots de 64 bits pour coder les flottants.
 - ☒ C'est parce que 0,1, 0,2 et 0,3 n'ont pas de représentation exacte en virgule flottante.
 - ☐ C'est une erreur de la machine.
 - ☐ C'est parce que $>$ signifie 'supérieur ou égal' pour l'interpréteur Python.
8. Soit le nombre x qui a pour écriture binaire 110,11011. Quelle est sa mantisse ?
- ☐ 110
 - ☒ 1,1011011
 - ☐ 1011011
 - ☐ 10
9. Lorsqu'on travaille avec des nombres à virgule flottante écrits sur 64 bits, à quoi faut-il faire attention ?
- ☐ Les nombres doivent rester entre -1,9e99 et 1,9e99 environ.
 - ☐ Les nombres doivent rester positifs.
 - ☒ Les nombres doivent rester entre -1,7e308 et 1,7e308 environ.
 - ☐ Il n'y a aucune limitation.
10. Parmi les systèmes d'exploitation suivants, lequel est libre (c'est à dire que l'utilisation, l'étude, la modification et la duplication en vue de sa diffusion sont permises) ?
- ☐ mac OS
 - ☒ GNU/Linux Debian
 - ☐ Microsoft Windows
 - ☐ Apple Unix A/UX
11. Voici un programme Python :
- ```
f = open('fichier.txt', 'w')
f.write('Payé ')
f.close()
f = open('fichier.txt', 'w')
f.write('2 €')
f.close()
```
- Que contient fichier.txt après l'exécution de ce programme ?
- ☐ Payé
  - ☐ Payé 2 €
  - ☒ 2 €
  - ☐ Payé  
2 €
12. Pour représenter la chaîne de caractères 'Payé 2 €' on peut utiliser la table de caractères
- ☐ Latin-1
  - ☐ ASCII
  - ☐ ISO8859-1
  - ☒ UTF-8

La table de caractères 'Latin-1' donne la correspondance **nombre hexadécimal / caractère** :

|   | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|
| 2 |   | ! | " | # | \$ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D  | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T  | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d  | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t  | u | v | w | x | y | z | { | } | ~ |   |   |
| 8 |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 9 |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| A |   | ı | ç | £ | ¤  | ¥ | ¦ | § | ¨ | © | ª | « | ¬ | - | ® | ¯ |
| B | ° | ± | ² | ³ | ´  | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| C | À | Á | Â | Ã | Ä  | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| D | Ð | Ñ | Ò | Ó | Ô  | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| E | à | á | â | ã | ä  | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| F | ð | ñ | ò | ó | ô  | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

Par exemple le nombre *hexadécimal* 41 correspond à 'A', donc le nombre *décimal* 65 correspond à 'A'.

13. La fonction intégrée dans Python chr(n) retourne le caractère en fonction de l'entier n dans la table de caractères latin-1. On peut donner la valeur de n en écriture décimale ou en écriture hexadécimale.

Par exemple chr(65) et chr(0x41) retournent 'A'.

Que va afficher chr(42) ?

- ☐ 'B'
- ☐ 'f'
- ☐ 'ê'
- ☐ **«\*»**

14. La fonction intégrée dans Python ord(caractere) retourne l'écriture décimale de l'entier n correspondant selon la table de caractères latin-1. Que va afficher ord('é') ?

- ☐ 41
- ☐ **233**
- ☐ e9
- ☐ Une erreur

15. Quel est le résultat de

'Payé'.encode('latin-1') ?

- ☐ **b'Pay\xe9'**
- ☐ b'Payé'
- ☐ b'Paye'
- ☐ b'Pay\xc3\xa9'

16. La suite d'octets 50 61 79 c3 a9 a été relevée dans un fichier texte codé en UTF-8. Que va afficher la commande b'\x50\x61\x79\xc3\xa9'.decode('latin-1') ?

- ☐ 'Payé'
- ☐ 'xP\xaxylxé'
- ☐ **'PayÃ©'**
- ☐ Une erreur

|                                                        |    |        |        |       |      |
|--------------------------------------------------------|----|--------|--------|-------|------|
| 17. Le fichier fichier.txt contient le texte suivant : | id | nom    | prenom | genre | date |
|                                                        | 1  | Turing | Alan   | M     | 1942 |
|                                                        | 2  | Borg   | Anita  | F     | 1969 |
|                                                        | 3  | Page   | Lary   | M     | 1998 |

Que faut-il écrire à la place des pointillés dans la boucle :

```
liste = []
f = open('fichier.txt', 'r')
for ligne in f:
 ...
```

pour obtenir liste = [['id', 'nom', 'prenom', 'genre', 'date'], ['1', 'Turing', 'Alan', 'M', '1942'], ['2', 'Borg', 'Anita', 'F', '1969'], ['3', 'Page', 'Lary', 'M', '1998']] ?

- ☒ liste.append(ligne.rstrip().split("\t"))
- ☐ liste.append(ligne.rstrip())
- ☐ liste.append(ligne)
- ☐ liste.append(ligne.split("\t"))

18. Qui a énoncé en 1945 les principes de l'architecture des ordinateurs ?

- ☐ Charles Babbage
- ☒ John Von Neumann
- ☐ Alan Turing
- ☐ Blaise Pascal

19. En Python, les programmes avant d'être interprétés sont pré compilés en 'bytecode'. Voici une séquence d'instructions simples en bytecode :

```
>>> import dis
>>> dis.dis(f)
11 0 LOAD_CONST 1 (2)
 2 STORE_FAST 0 (a)
12 4 LOAD_CONST 2 (3)
 6 STORE_FAST 1 (b)
13 8 LOAD_FAST 0 (a)
 10 LOAD_FAST 1 (b)
 12 BINARY_ADD
 14 RETURN_VALUE
```

A quel programme en langage Python correspond-elle ?

- ☒

```
def f():
 a = 2
 b = 3
 return a + b
```
- ☐

```
def f():
 a = 2
 b = 3
 return a - b
```
- ☐

```
def f():
 a = 3
 b = 2
 return a * b
```
- ☐

```
def f():
 a = 3
 b = 2
 return a + b
```

20. Alan a reçu la liste cryptée suivante : 114 104 102 109 100 115

Quel est le texte 'mon\_message' d'origine sachant que les codes Python qui ont servi au cryptage sont :

```
def codage_en_latin1(texte):
 """
 Retourne la liste des entiers correspondant aux caracteres de 'texte'.

 Parametres nommes
 texte : de type str
 Chaîne de caracteres en clair. Exemple : texte = 'Dé'

 Retourne
 liste : de type list
 Liste d'entiers ecrits comme des chaines de caractères contenant
 l'écriture hexadecimale. Exemple : liste = ['0x44', '0xe9']
 """
 liste = []
 octets = texte.encode('latin-1') # Ex. texte = 'Dé' donc octets = b'D\xe9'
 for elt in octets:
 liste.append(hex(elt)) # Conversion de 'D' en '0x44' et de '\xe9' en '0xe9'
 return liste # Exemple : liste = ['0x44', '0xe9']

def cryptage(liste):
 """
 Modifie les nombres de la liste donnee en argument

 Parametres nommes
 liste = de type list
 Liste d'entiers sous forme de chaines de caracteres. Ex.['0x44', '0xe9'].

 Retourne
 Liste_chiffree : de type list
 Liste d'entiers en ecriture decimale. Ex. [67, 232].
 """
 liste_cryptee = []
 for elt in liste:
 elt_decimal = int(elt, 16) # Conversion. Ex '0x44' devient 68.
 elt_crypte = elt_decimal - 1 # Cryptage par soustraction. 68 devient 67.
 liste_cryptee.append(elt_crypte)
 return liste_cryptee # Exemple : liste_cryptee = [67, 232]

mon_message = '??????'
ma_liste = codage_en_latin1(mon_message)
ma_liste_cryptee = cryptage(ma_liste)
print(ma_liste_cryptee)
```

- ☐ saisie
- ☐ server
- ☐ **signet**
- ☐ smiley