DIU NSI - QCM – Blocs 1 - 2

*Le corrigé est en texte masqué*

**Représentation des données : types et valeurs de base**

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Écriture d’un entier positif dans une base b ⩾ 2 | Passer de la représentation d’une base dans une autre. | Les bases 2, 10 et 16 sont privilégiées. |

1. Sur un octet (8 bits), le nombre 153 s’écrit en binaire (base 2) 128+0+0+16+8+0+0+1

⭘ 10011011 ⭘ 10011101 🞊 ⭘ 10011001

1. Le nombre entier positif codé en binaire par 01011010 (base 2) est : 0+64+0+16+8+0+2+0

⭘ 80 ⭘ 84 🞊 ⭘ 90 ⭘ 100

1. Le nombre 19 en base 10 s’écrit en hexadécimal 1x161 + 3x160

⭘ 4F 🞊 ⭘ 13 ⭘ 19 ⭘ F4

1. Le nombre hexadécimal B42 s’écrit en base 10 11x16x16 + 4x16 + 2 = 2882

⭘ 1142 🞊 ⭘ 2882 ⭘ 242

1. Le nombre binaire (base 2) 01111010 (122) s’écrit en base 16 01111 = 7 et 1010 = 10 (A)

🞊 ⭘ 7A ⭘ A7 ⭘ 32 ⭘ 2 ⭘ 7F

1. Le nombre hexadécimal E9 (233) s’écrit en binaire (base 2) E = 14 = 1110 et 9 = 1001

🞊 ⭘ 11011001 ⭘ 11101011 🞊 ⭘ 11101001

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Représentation binaire d’un entier relatif | Évaluer le nombre de bits nécessaires à l’écriture en base 2 d’un entier, de la somme ou du produit de deux nombres entiers.  Utiliser le complément à 2. | Il s’agit de décrire les tailles courantes des entiers (8, 16, 32 ou 64 bits).  Il est possible d’évoquer la représentation des entiers de taille arbitraire de Python. |

1. La somme des nombres binaires 01010101 (85) et 10010010 (146) 85+146 = 231 = 128+64+32+0+0+4+2+1

🞊 ⭘ 11100111 ⭘ 11110111 ⭘ 111101011

1. Le produit des nombres binaires 0101 (6) et 0010 (2) 6 x 2 = 12

🞊 ⭘ 0111 ⭘ 0110 ⭘ 1100

1. Les entiers signés sur 7 bits s’étendent :

⭘ de - 128 à 128 ⭘ de - 127 à 128 🞊 ⭘ de - 128 à 127 ⭘ de - 127 à 127

1. Déterminer la valeur de l’entier signé 1 0101110 (bit de gauche à 1 pour le négatif) en utilisant le complément à 2 :

⭘ -46 ⭘ - 83 ⭘ - 81 ⭘ - 82- 128-46

1. Déterminer l’écriture binaire l’entier signé -100 (bit de gauche à 1 pour le négatif) en utilisant le complément à 2 :

⭘ 1 1101000 🞊 ⭘ 1 0011100 ⭘ 1 0011011

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Représentation approximative des nombres réels : notion de nombre flottant | Calculer sur quelques exemples la représentation de nombres réels : 0.1, 0.25 ou1/3. | 0.2 + 0.1 n’est pas égal à 0.3.  Il faut éviter de tester l’égalité de deux flottants.  Aucune connaissance précise de la norme IEEE-754 n’est exigible. |

Considérons un nombre flottant en 8 bits (1 pour le signe, 2 pour l’exposant et 5 pour la mantisse)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| signe | exposant | | mantisse | | | | |  |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |  |
| - | exposant : -3 | | valeur codée : 23 (complément à 32 : 9) | | | | | soit - -9 x 10-3 = - 0,009 |

1. Selon le même modèle, le nombre flottant 0 | 10 | 11001 vaut

⭘ 2,5 ⭘ -2,5 ⭘ 0,27 ⭘ -0,27 🞊 ⭘ 0,25 ⭘ -0,25 ⭘ 0,027 ⭘ -0,027

Considérons un nombre flottant en 24 bits (1 pour le signe, 4 pour l’exposant et 19 pour la mantisse)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| signe | exposant | | | | mantisse | | | | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| + | ? | | | | valeur : 333 333 | | | | | | | | | | | | | | | | | | |

1. Ce nombre flottant vaut

🞊 ⭘ 0,333333 ⭘ 1/3 ⭘ 3,33333 ⭘ -0,0333333

1. Combien de décimales peut-on obtenir au maximum avec ce format de nombre flottant ?

⭘ une infinité 🞊 ⭘ 14 ⭘ 15 ⭘ 16

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Valeurs booléennes : 0, 1. Opérateurs booléens :  and, or, not.  Expressions booléennes | Dresser la table d’une expression booléenne. | Le ou exclusif (xor) est évoqué.  Quelques applications directes comme l’addition binaire sont présentées.  L’attention des élèves est attirée sur le caractère séquentiel de certains opérateurs booléens. |

1. a = 0 et b = 1 a and b vaut 🞊 ⭘ 0 ⭘ 1
2. a = 1 et b = 1 a and b vaut ⭘ 0 🞊 ⭘ 1
3. a = 0 et b = 0 a or b vaut 🞊 ⭘ 0 ⭘ 1
4. a = 1 et b = 0 a or b vaut ⭘ 0 🞊 ⭘ 1
5. a = 1 et b = 0 a or not b vaut ⭘ 0 🞊 ⭘ 1
6. a = 1 et b = 0 not a or b vaut 🞊 ⭘ 0 ⭘ 1
7. a = 1 et b = 0 a and not b vaut ⭘ 0 🞊 ⭘ 1
8. a = 0, b = 0 et c = 1 a and b or c vaut ⭘ 0 🞊 ⭘ 1
9. a = 1, b = 0 et c = 1 a and (b or c) vaut ⭘ 0 🞊 ⭘ 1
10. a = 1, b = 1 et c = 0 (a and b) or c vaut 🞊 ⭘ 0 ⭘ 1

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Représentation d’un texte en machine.  Exemples des encodages ASCII, ISO-8859-1, Unicode | Identifier l’intérêt des différents systèmes d’encodage.  Convertir un fichier texte dans différents formats d’encodage. | Aucune connaissance précise des normes d’encodage n’est exigible |

1. Le nombre de caractères codable en binaire est fonction du nombre de bits utilisés 🞊 ⭘ vrai ⭘ faux
2. Le codage des lettres en ASCII prend en compte plus d’alphabets que l’UTF-8 ⭘ vrai 🞊 ⭘ faux
3. UTF-8 utilise systématiquement 4 octets ⭘ vrai 🞊 ⭘ faux
4. UTF-8 est compatibles ASCII 🞊 ⭘ vrai ⭘ faux
5. Certains caractères sont traités plus « vite » / plus « légers » que d’autres en UTF-8 🞊 ⭘ vrai ⭘ faux
6. La taille en octets d’un fichier texte en UTF-8 détermine le nombre de caractères ⭘ vrai 🞊 ⭘ faux

**Représentation des données : types construits**

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| p-uplets.  p-uplets nommés | Écrire une fonction renvoyant un p-uplet de valeurs. |  |

Soit le programme suivant :

def myst(a,b):

a = a - 1

b = b / a

b = a + b

return b, a, a-b

myst(4,6)

1. Déterminer le p-uplet retourné :

⭘ (4, 6, -2) ⭘ (3, 5, 2) 🞊 ⭘ (5, 3, -2)

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Tableau indexé, tableau donné en compréhension | Lire et modifier les éléments d’un tableau grâce à leurs index.  Construire un tableau par compréhension.  Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i] [j].  Itérer sur les éléments d’un tableau. | Seuls les tableaux dont les éléments sont du même type sont présentés.  Aucune connaissance des tranches (slices) n’est exigible.  L’aspect dynamique des tableaux de Python n’est pas évoqué.  Python identifie listes et tableaux.  Il n’est pas fait référence aux tableaux de la bibliothèque NumPy. |

Soit la fonction suivante :

def t:

liste = []

for i in range(3):

liste.append(i\*2)

return liste

1. Déterminer la liste retournée :

⭘ [2, 4, 6] 🞊 ⭘ [0, 2, 4] ⭘ [0, 2, 4, 6]

Soit le tableau (liste) t = [1, 0, 1, 8, 6]

1. Quelle est la valeur de t[1] ? 🞊 ⭘ 0 ⭘ 1 ⭘ 2
2. Quelle est la longueur de t ? 🞊 ⭘ 5 ⭘ 6 ⭘ 8
3. Quelle est l’indice de la valeur 6 ? 🞊 ⭘ 4 ⭘ 5 ⭘ 6

Soit le tableau (liste) t = [ ['a', 'b'], ['c', 'd', 'e'], 'z']

1. Comment atteindre la valeur 'e' ? ⭘ t[2][3] ⭘ t[3][2] 🞊 ⭘ t[1][2] ⭘ t[2][1]
2. Comment atteindre la valeur 'z' ? ⭘ t[2] ⭘ t[3] 🞊 ⭘ t[2][0] ⭘ t[3][0]
3. Que donne l = 5\*[0] ? 🞊 ⭘ 0 ⭘ [0, 0, 0, 0, 0] ⭘ [5, 5, 5, 5, 50]

Soit le code suivant :

l = 2\*[0]

for i in range(len(l)):

l[i] =3 \*[i]

1. Que donne ce programme ?

🞊 ⭘ [ [0, 0, 0], [1, 1, 1] ] ⭘ [ [0, 0, 0], [3, 3, 3] ] ⭘ [ [1, 1, 1], [3, 3, 3] ] ⭘ [ [0, 0], [1, 1], [3, 3] ]

Soit le code suivant :

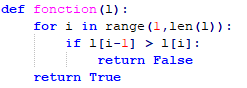
l = [[0 for i in range(8)] for j in range(10)]

1. Que donne print(len(l)) ?

⭘ 80 ⭘ 8 🞊 ⭘ 10

1. l est un tableau de tableaux (matrice). Quel(s) programme(s) Python permet(tent) d’appliquer l’action sur touts ses éléments ?

|  |  |  |
| --- | --- | --- |
| 🞎 | 🗹🞎 | 🞎 |
| for i in range(len(t)):  for j in range(len(t[i])):  # action sur t[i][j] | for i in range(len(t)):  for j in range(len(t[i])):  # action sur t[i][j] | for i in range(len(t)):  for j in range(len(t[i])):  # action sur t[j][i] |
| 🞎 | 🞎 | 🗹🞎 |
| for i in range(len(t[i])):  for j in range(len(t[j])):  # action sur t[i][j] | for i in range(len(t)):  for j in range(len(t)):  # action sur t[i][j] | for j in range(len(t)):  for i in range(len(t[j])):  # action sur t[j][i] |

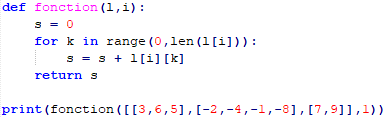
1. l étant une liste, la fonction ci-contre permet de retourner (l étant une liste) :

⭘ a. Une information sur le maximum d’une liste.

⭘ b. Une information sur le minimum d’une liste.

⭘ c. Une information sur la présence d’occurrences

🞊 ⭘ d. Une information sur le tri de la liste

1. La fonction ci-contre retourne :

⭘ a. 8

⭘ b. 14

🞊 ⭘ c. -15

⭘ d. -2

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Dictionnaires par clés et valeurs | Construire une entrée de dictionnaire.  Itérer sur les éléments d’un dictionnaire. | Il est possible de présenter les données EXIF d’une image sous la forme d’un enregistrement.  En Python, les p-uplets nommés sont implémentés par des dictionnaires.  Utiliser les méthodes keys(),values () et items (). |

Soit le dictionnaire suivant : dict = {'France': 'Paris', 'Espagne': 'Madrid'}

1. Quels programmes Python permettent d’afficher la liste des capitales ?

|  |  |  |
| --- | --- | --- |
| 🞎 | 🗹🞎 | 🞎 |
| for i in dict.keys():  print(i) | for i in dict.values():  print(i) | for i in range(len(dict)):  print dict[i] |
| 🗹🞎 | 🗹🞎 |  |
| print (dict.values()) | for i in dict.keys():  print(dict[i]) |  |

1. Quelle est l’instruction exacte pour ajouter un couple de valeurs ?

⭘ dict = {'Italie': 'Rome'} 🞊 ⭘ dict['Italie'] = 'Rome' ⭘ dict['Italie': 'Rome']

**Langages et programmation**

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Constructions élémentaires | Mettre en évidence un corpus de constructions élémentaires. | Séquences, affectation, conditionnelles, boucles bornées, boucles non bornées, appels de fonction. |

1. Une variable

🗹 🞎 a. peut changer de valeur d’une exécution à l’autre d’un programme

🗹 🞎 b. peut changer de valeur au cours de l’exécution d’un programme

🗹 🞎 c. permet d’appliquer un même traitement à différentes valeurs (lors de différentes exécutions du programme)

🞎 d. ne peut être utilisée que pour des valeurs de type numérique

🞎 e. ne peut concerner que des valeurs utilisées dans des calculs

1. Laquelle des instructions (d’affectation) suivantes est inexacte ?

🞎 a. var = 2 🞎 b. var = "hello !" 🞎 c. var3 = ""

🗹 🞎 d. var = bonjour ! 🞎 e. var5 = False 🞎 f. var1 = var2

1. a et b sont des variables déclarées. Voici la seule instruction de départ : a=1

Parmi les instructions suivantes (indépendantes et non successives, c’est-à-dire ayant chacune lieu après l’instruction de départ), choisir la ou les instruction(s) exacte(s), c’est-à-dire ne provoquant pas d’erreur d’exécution :

🗹 🞎 b = a 🗹 🞎 a = a + 1 🞎 b = b + 1 🞎 a = b

1. Les instructions d'affectation a=b et b=a sont équivalentes :

⭘ vrai 🞊 ⭘ faux

1. Associer chacun des termes à leur définition (en indiquant la lettre à associer) :

affectation : c a. objet stockant une valeur inchangée d’une exécution à une autre du même programme

indentation : d b emplacement en mémoire de stockage d’une valeur

programme : f c. action de stocker une valeur dans une variable

variable : b d. convention de présentation aidant à repérer la structure du code

constante : a e. suite d’opérations mettant en œuvre un raisonnement afin de réaliser un traitement

compilation : g f. réalisation d’un algorithme par un ordinateur

algorithme : e g. transcription du code en langage compréhensible par la machine

1. Les conditions a= =b et b= =a sont équivalentes :

🞊 ⭘ vrai ⭘ faux

1. La mise en place d’une structure alternative

🞎 a. permet de répéter une séquence d’instructions

🗹 🞎 b. permet de traiter plusieurs cas dans un seul programme plutôt qu’avec plusieurs programmes

🗹 🞎 c. repose sur une condition booléenne

🞎 d. conduit toujours à développer plusieurs branches (if… else)

1. Toutes les lignes d’instructions d’une structure alternative (if… else) sont exécutées lors d’une instance de programme.

⭘ vrai 🞊 ⭘ faux

1. La variable pnom contient un prénom.

ne peut être exécutée délivre un résultat aucune des ces

en raison d’une erreur satisfaisant deux réponses

L’instruction Python print("Bonjour pnom") 🞎 🞎 🗹 🞎

L’instruction Python print("Bonjour " pnom) 🗹 🞎 🞎 🞎

L’instruction Python print("Bonjour ", pnom) 🞎 🗹 🞎 🞎

1. Un jeu d'essai est un ensemble de groupes de valeurs pour les variables d’un programme (chaque groupe associe une valeur à une variable et permet de tester une fois le programme). Un jeu d'essai permet de tester plusieurs fois un programme et ainsi de détecter deux des quatre types d'erreurs « possibles » en programmation :

🞎 a. erreur de syntaxe instructions mal écrites, …

🞎 b. erreur de compilation structures non « fermées », …

🗹 🞎 c. erreur d'exécution le programme « plante »

🗹 🞎 d. erreur de logique le programme ne « plante » pas mais délivre un résultat faux

1. On peut imbriquer plusieurs structures alternatives pour traiter plus de deux cas (elif, équivalent à un if dans le else). Généralement pour traiter *n* cas, il faut en imbriquer :

⭘ n ⭘ n+1 🞊 ⭘ n-1 ⭘ n-2 ⭘ n/2

1. Choisir la ou les affirmation(s) exacte(s) :

🗹 🞎 a. une structure itérative permet de répéter une séquence d’instructions

🞎 b. une structure itérative permet de choisir automatiquement une séquence d’instructions parmi deux

🗹 🞎 c. une structure itérative peut imbriquer d’autres structures (itératives ou alternatives)

🗹 🞎 d. une structure itérative peut être imbriquée dans d’autres structures (itératives ou alternatives)

🞎 e. une structure itérative peut être sans fin

1. On utilise une structure itérative avec compteur lorsqu’on connaît le nombre d’itérations (nombre de tours de la boucle)

⭘ a. dès le début du programme

🞊 ⭘ b. avant le premier tour de boucle

⭘ c. seulement lorsqu’on quitte la boucle

1. On utilise une structure itérative sans compteur lorsqu’on connaît le nombre d’itérations (nombre de tours de la boucle)

⭘ a. dès le début du programme

⭘ b. avant le premier tour de boucle

🞊 ⭘ c. seulement lorsqu’on quitte la boucle

1. La structure itérative la plus générale (celle permettant de programmer l’autre) est :

⭘ a. la structure avec compteur (For)

🞊 ⭘ b. la structure sans compteur Tant que (While)

1. Choisir la ou les affirmation(s) exacte(s) :

🗹 🞎 a une fonction est un sous-programme appelé par un programme principal

🗹 🞎 b. une fonction peut comporter aucun à plusieurs paramètres en entrée

🗹 🞎 d. un développeur peut créer ses propres fonctions

🗹 🞎 e. les langages sont "livrés" avec des fonctions prédéfinies

🞎 f. l'ordre dans lequel les arguments sont saisis lors de l'appel d'une fonction n'est pas important

1. Déterminer la valeur de a en fin de programme (ci-dessous) :

⭘ erreur ⭘ 4 🞊 ⭘ 8 ⭘ 10

def calcul(x,y): #fonction

{

return 2x + y

}

a = 4

if calcul(a,2)>10:

a = calcul(a,2)

else:

a = calcul(2,a)

1. L'opération qui consiste à introduire des décalages à droite dans le code :

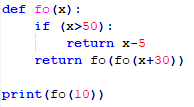
🗹 🞎 a. se nomme indentation

🞎 b. se nomme identation

🗹 🞎 c. aide le développeur à lire le code

🞎 d. aide l'utilisateur final de l'application l'utilisateur ne voit pas le code

🗹 🞎 e. est obligatoire dans certains langages

1. Quelle la sortie du code ci-contre :

⭘ a. 10

🞊 ⭘ b. 50

⭘ c. 45

⭘ d. 55

**Algorithmique**

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Parcours séquentiel d’un tableau | Écrire un algorithme de recherche d’une occurrence sur des valeurs de type quelconque.  Écrire un algorithme de recherche d’un extremum, de calcul d’une moyenne. | On montre que le coût est linéaire. |

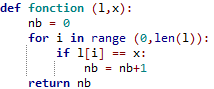
1. La fonction ci-dessous est appliquée à la liste l=[3,7,-4,7,9] et x=7 ; la valeur retournée est :

⭘ a. nb=7

⭘ b. nb=1

🞊 ⭘ c. nb=2

⭘ d. nb=-4



1. Pour la fonction ci-dessous, le coût est :

⭘ a. constant

🞊 ⭘ b. linéaire

⭘ c. logarithmique

⭘ d. quadratique



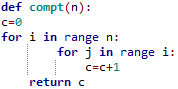
1. Quelle est la complexité temporelle du code ci-dessous ?

⭘ a. O(n)

🞊 ⭘ b. O(n2)

⭘ c. O(n\*Logn)

⭘ d. O(nLognLogn)



|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Tris par insertion, par sélection | Écrire un algorithme de tri.  Décrire un invariant de boucle qui prouve la correction des tris par insertion, par sélection. | La terminaison de ces algorithmes est à justifier.  On montre que leur coût est quadratique dans le pire cas. |

1. Soit la liste suivante T=[9,6,1,4,8] triée par insertion, sa représentation après la deuxième itération est :

⭘ a. T=[1,4,8,9,6]

⭘ b. T=[1,4,9,6,8]

🞊 ⭘ c. T=[1,6,9,4,8] [9, 6, 1, 4, 8] → [6, 9, 1, 4, 8] → [1, 6, 9, 4, 8]

⭘ d. T=[1,4,6,8,9]

1. Soit la liste suivante T=[5,1,2,7,6,4,3] triée par sélection, sa représentation après la troisième itération est :

⭘ a. T=[1,2,3,4,5,6,7]

⭘ b. T=[1,2,5,7,6,4,3]

⭘ c. T=[1,2,3,5,7,6,4]

⭘ d. T=[1,2,3,7,6,4,5] [5, 1, 2, 7, 6, 4, 3] → [1, 5, 2, 7, 6, 4, 3] → [1, 2, 5, 7, 6, 4, 3] → [1, 2, 3, 7, 6, 4, 5]

1. Un invariant de boucle sert à montrer :

⭘ a. La terminaison d’un algorithme récursif ä

🞊 ⭘ b. La validité d’un algorithme itératif

⭘ c. La complexité d’un algorithme itératif

⭘ d. La complexité d’un algorithme récursif

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Algorithme des k plus proches voisins | Écrire un algorithme qui prédit la classe d’un élément en fonction de la classe majoritaire de ses k plus proches voisins. | Il s’agit d’un exemple d’algorithme d’apprentissage. |

Méthode des k plus proches voisins (kPPV)

Dans la figure ci-dessous, les points représentent un ensemble de vecteurs de dimension 2, appartenant à 2 classes appelées A et B. L’ordre de sélection des vecteurs est indiqué par les indices situés à côté de chacun. Les points 1 à 4 sont déjà classés ; on applique donc l’algorithme en commençant avec le point 5.

1. Appliquer la méthode des kPPV avec k=3. La classe résultante de chaque point est :

🞊 ⭘ a. 5(B), 6(A), 7(B), 8(A), 9(A), 10(A)

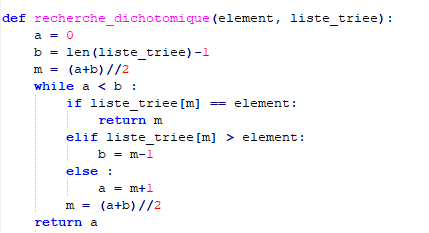
⭘ b. 5(B), 6(A), 7(B), 8(B), 9(A), 10(A)

⭘ c. 5(B), 6(A), 7(B), 8(A), 9(A), 10(B)

⭘ d. 5(B), 6(B), 7(B), 8(A), 9(A), 10(A)

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Recherche dichotomique dans un tableau trié | Montrer la terminaison de la recherche dichotomique à l’aide d’un variant de boucle. | Des assertions peuvent être utilisées.  La preuve de la correction peut être présentée par le professeur. |

Soit le programme de recherche dichotomique suivant :



1. Un variant permettant de montrer la terminaison de cette recherche est :

⭘ a. m

⭘ b. element

⭘ c. b

🞊 ⭘ d. b-a

**Interactions entre l’homme et la machine**

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Modalités de l’interaction entre l’homme et la machine  Événements | Identifier les différents composants graphiques permettant d’interagir avec une application Web.  Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter. | Il s’agit d’examiner le code HTML d’une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript. |
| Interaction avec l’utilisateur dans une page Web | Analyser et modifier les méthodes exécutées lors d’un clic sur un bouton d’une page Web. |  |

1. Client serveur

🞎 a. sont des notions avant tout matérielles

🗹 🞎 b. sont des notions avant tout logicielles

🗹 🞎 c. un même poste peut être à la fois client et serveur

🗹 🞎 d. un même poste peut exécuter plusieurs serveurs

🞎 e. un même poste ne peut exécuter qu’un seul client

1. Le HTML

⭘ a. met en forme du contenu

🞊 ⭘ b. structure du contenu

1. Le langage HTML est

⭘ la propriété du W3C

🞊 ⭘ normalisé par le W3C

⭘ reconnu uniquement dans sa version 5 il aurait fallu réécrire tous les anciens sites

1. Choisir l’affirmation inexacte à propos des fichiers HTML

⭘ ils sont interprétés par les navigateurs

⭘ ils sont au format texte

🞊 ⭘ ils sont stockés par le navigateur stockés en local, ils sont consultables sans HTTP mais ne sont pas stockés par le navigateur

⭘ ils peuvent être stockés par le serveur web et transmis via HTTP

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Formulaire d’une page Web | Analyser le fonctionnement d’un formulaire simple.  Distinguer les transmissions de paramètres par les requêtes POST ou GET. | Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité. |

1. Quelle action de formulaire permet de transférer plus de 255 caractères ?

⭘ a. APPEND

🞊 ⭘ b. POST

⭘ c. GET

⭘ d. SEND

1. La méthode POST est une méthode de transmission des données

⭘ a. moins sûre que la méthode GET

🞊 ⭘ b. plus sûre que la méthode GET les données ne sont pas stockées dans l’historique des URL

⭘ c. totalement sûre les données circulent dans un paquet à part mais en clair (sauf si HTTPS)

⭘ d. souvent utilisée pour transmettre des mots clés de recherche GET permet de refaire la même recherche en copiant-collant l’URL avec les paramètres de recherche inclus

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Interaction client-serveur.  Requêtes HTTP, réponses du serveur | Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre.  Distinguer ce qui est mémorisé dans le client et retransmis au serveur.  Reconnaître quand et pourquoi la transmission est chiffrée. | Il s’agit de faire le lien avec ce qui a été vu en classe de seconde et d’expliquer comment on peut passer des paramètres à un site grâce au protocole HTTP. |

1. Dans une architecture client/serveur, trouver l’affirmation fausse (les clients ne pouvant pas fonctionner en serveur) :

⭘ a. Il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modifications majeures.

⭘ b. Le serveur peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance.

🞊 ⭘ c. Les « clients » peuvent fonctionner entre eux de manière autonomes.

⭘ d. Le client met en forme le contenu et l’affiche dans la fenêtre.

1. Dans l’architecture Client/serveur (Web), lors d’une requête http le serveur Web renvoie :

⭘ a. l’adresse IP (du serveur Web)

⭘ b. le nom de domaine (du serveur Web)

🞊 ⭘ c. le document HTML (page Web)

⭘ d. toutes ces réponses sont correctes

|  |  |  |
| --- | --- | --- |
| **Contenus** | **Capacités attendues** | **Commentaires** |
| Événements clés de l’histoire de l’informatique | Situer dans le temps les principaux événements de l’histoire de l’informatique et leurs protagonistes. | Ces repères historiques seront construits au fur et à mesure de la présentation des concepts et techniques |

1. En quelle année a été inventé le HTML ?

⭘ a. 1970

⭘ b. 1980

🞊 ⭘ c. 1990

⭘ d. 2000

1. Choisir l’affirmation exacte :

⭘ a. le web est entièrement accessible deep web (parties privées) et dark web (non indexé)

⭘ b. les applications des ordiphones font partie du web consultable à travers un navigateur

🞊 ⭘ c. le web est une application d’internet

⭘ d. web et internet sont synonymes