Exercices de travaux dirigés d'algorithmique / programmation Javascript

Exercice 1 – Petite trace (C1a, C1b)

Quelles seront les valeurs des variables *a* et *b* à la fin de l'algorithme **algo1** ? Faites une trace !

Algorithme **algo1**
Variables d'entrée : -
Variables : entiers *a* et *b*
Type de sortie : -
Début
    a ← 1
    b ← *a*+3
    a ← 3
Fin

Trace :
valeurs des variables après chaque ligne de l'algorithme

ligne	a	b
0		
1		
2		
3		

Exercice 2 – Petite trace avec entrées (C1a, C1b)

Quel est le résultat de l'algorithme **algo2**(3,5) ? Faites une trace !

Algorithme **algo2**
Variables d'entrée : entiers *a* et *b*
Variables : entier *c*
Type de sortie : entier
Début
    c ← *a*+*b*
    a ← 2
    c ← *b*-*a*
    renvoyer *c*
Fin

ligne	a	b	c
0			
1			
2			
3			

Exercice 3 – Traitement d'entiers (C1b, C2b, C4a)

Écrivez en Javascript et en pseudo-code les instructions suivantes, et faites-en la trace.
1. J'initialise *a* à 8 2. Je stocke *a*+1 dans *b* 3. J'ajoute 1 à *a* 4. Je multiplie *a* par 2 5. Je retranche 5 à *b*
6. Je multiplie *b* par lui-même 7. J'ajoute 3 à 2 8. Si *a* vaut 1, je lui ajoute 2, sinon je lui ajoute 3

Exercice 4 – Traitement de chaînes de caractères (C1b, C4a)

Écrivez en Javascript les instructions suivantes. Quelle est la valeur de *a* à la fin de ces instructions ?
1. Je mets les chaînes de caractères "i", "an", "g", "ory", "the" dans les variables respectives *a*, *b*, *c*, *d*, *e*.
2. Je mets la chaîne *e* suivie de la chaîne *d* dans la chaîne *d*. 3. J'insère la chaîne "b" au début des chaînes *a* et *b*.
4. J'ajoute la chaîne *c* à la fin des chaînes *a* et *b* 5. Je mets dans la chaîne *a* la chaîne *e* suivie de *a* suivie de *b* suivie de *d*.

Exercice 5 – Échange de valeurs de variables (C2a, C2b, C4a)

Écrivez en pseudo-code et en Javascript un algorithme qui prend en entrée deux variables entières *a* et *b* et échange leurs valeurs.

Exercice 6 – Conjecture de Syracuse

Q1 (C2a, C2b, C4a). Dessinez l'organigramme, puis écrivez le code en Javascript, de l'algorithme **syracuse** qui prend en entrée un entier *a*>0 puis :
1. Si *a* est différent de 1 :
  - s'il est pair, on le divise par deux puis on revient à l'étape 1 ;
  - s'il est impair, on le multiplie par 3 puis on lui ajoute 1, et on revient à l'étape 1 ;
2. Sinon, on arrête le programme

Q2 (C1b). Faites la trace du programme en choisissant en entrée le nombre de lettres de votre prénom (pour les prénoms composés, limitez-vous au premier...).

En Javascript, le reste dans la division euclidienne de *a* par *b* s'écrit : `a%b`
a est un nombre pair si le reste dans la division euclidienne de *a* par 2 est 0

Exercice 7 – Conversions et écriture en binaire

- **Q1 (C1b).** À quel nombre (en décimal) correspond le nombre suivant en binaire : 10010110 ?
- **Q2 (C1b).** Écrivez 42 en binaire
- **Q3 (C1b).** Écrivez 84 en binaire
- **Q4 (C2a).** Que remarquez-vous ?
- **Q5 (C2a, C2b, C4a).** Déduisez-en un algorithme en Javascript **diviseParDeuxPair** qui prend en entrée une chaîne de caractères qui contient un nombre binaire *n* et s'il est pair, renvoie une chaîne de caractères qui contient la valeur de *n*/2 écrite en binaire.

Vous utiliserez l'algorithme **caractère** qui prend en entrée une chaîne de caractères *chaine* et un entier *i* et renvoie le *i*-ième caractère de *chaine*, ainsi que l'algorithme **sousChaine** qui renvoie la partie de la chaîne de caractères *chaine* allant du *i*-ième au *j*-ième caractère (inclus), et l'algorithme **longueur** qui renvoie le nombre de caractères de la chaîne de caractères *chaine*.

Exercice 8 – Algorithmes pour traiter des textes

Le but de cet exercice est de manipuler en Javascript les algorithmes de traitement des chaînes de caractères suivants :
- **`longueur`**`(a)` renvoie le nombre de caractères de la chaîne de caractères `a`.
- **`sousChaine`**`(a,i,j)` renvoie la partie de la chaîne de caractères `a` comprise entre le `i`-ième caractère et le `j`-ième caractère inclus.
- **`gauche`**`(a,i)` renvoie les `i` premiers caractères de la chaîne de caractères `a`.
- **`droite`**`(a,i)` renvoie les `i` derniers caractères de la chaîne de caractères `a`.

Q1 (C1c). Faites un exemple d'appel de chacun de ces algorithmes, en écrivant le résultat obtenu en sortie à partir d'entrées que vous avez choisies librement. Par exemple, pour la première, `longueur("Na je")` renvoie 5.
Q2 (C1c). Quel est le type de sortie de chacun des algorithmes ?
Q3 (C1b, C3a). Que contient la variable `a` après les deux instructions suivantes en Javascript ?

```
var a = "Na je nun ta sa ro un in gan jo gin yo ja";
a = gauche(droite(a,5),2) + sousChaine(a,(longueur(a)+1)/2+5,27);
```

Q4 (C1b, C3a). Que contient la variable `b` après les deux instructions suivantes en Javascript ?

```
var b = "Ko pi han ja ne yo yu rul a neun pum gyo gi nun yo ja";
b = gauche(droite(b,20),3)+droite(droite(droite(droite(droite(b,47),42),23),9),1);
```

Q5 (C2a, C2b, C4a). Supposons que l'algorithme **sousChaine** ne vous a pas été fourni. Écrivez-le en Javascript, en utilisant les algorithmes `gauche` et `droite` (trouvez sur un exemple l'idée à exploiter!).

Q6 (C1c, C2a). On souhaite écrire en Javascript un algorithme **premierMot**`(a)` qui renvoie le premier mot de `a`, c'est-à-dire la portion de la chaîne de caractères `a` qui précède le premier espace dans `a`.

Que renvoie **premierMot**`("Ba mi o")` ?

Q7 (C2a, C2b, C4a, C4b, un peu difficile). Écrivez en Javascript l'algorithme **premierMot**.

Q8 (C2a, C2b, C4a, C4b, difficile, pour s'entraîner chez soi). Écrivez un algorithme Javascript **mots**`(a)` qui renvoie un tableau de chaînes de caractères dont chaque case contient un mot de `a` (on considère que seuls les espaces séparent les mots), dans leur ordre d'apparition. Par exemple, **mots**`("Na nun sa na ye")` renvoie le tableau `["Na","nun","sa","na","ye"]`.

Exercice 9 – Manipulation de tableau (C1b, C1c)

Algorithme **mystere**
Variable d'entrée : un tableau *t* de chaînes de caractères
Variables : chaînes de caractères *a* et *b*
Type de sortie : chaîne de caractères
Début
1. *a* ← **concatene**(**case**(*t*,1),**case**(*t*,2))
2. *b* ← **concatene**(**case**(*t*,4),**case**(*t*,1))
3. **case**(*t*,1) ← **concatene**("b",**case**(*t*,1))
4. *a* ← **concatene**(**case**(*t*,6),*a*)
5. *a* ← **concatene**(**case**(*t*,1),*a*)
6. *b* ← **concatene**(**case**(*t*,1),*b*)
7. *a* ← **concatene**(*a*,**case**(*t*,3))
8. *b* ← **concatene**(**case**(*t*,5),*b*)
9. renvoyer **concatene**(*a*,*b*)

Fin

On rappelle que **concatene**(*a*,*b*) prend en entrée deux chaînes de caractères *a* et *b* et renvoie une chaîne de caractère contenant la chaîne *a* suivie de la chaîne *b*. Par exemple, **concatene**("a","b") renvoie la chaîne de caractères "ab".

Que renvoie l'algorithme **mystere**(["a","c","k","m","o","r"]) ? (faites une trace)

Exercice 10 – Traitement d'un tableau de notes

On a à disposition un tableau *t* d'entiers correspondant aux notes d'un étudiant : [15,9,10,17,8,14].

Q1 (C2a, C2b). Écrivez en pseudo-code un algorithme **cherche** qui prend en entrée un tableau d'entiers *t* ainsi qu'un entier *a* et renvoie le booléen VRAI si l'entier *a* est dans le tableau *t*, et FAUX sinon.

Q2 (C1b, C1c). Pour savoir si l'étudiant a obtenu un 18, on va donc appeler l'algorithme **cherche**([15,9,10,17,8,14], 18) : quelle valeur obtiendra-t-on en sortie ?

Q3 (C2a, C2b). Écrivez en pseudo-code un algorithme **somme** qui prend en entrée un tableau d'entiers *t* et qui renvoie la somme des entiers contenus dans *t*.

Q4 (C1b, C1c). Que renvoie **somme**([15,9,10,17,8,14]) ?

Q5 (C1a, C2b). En appelant l'algorithme **somme**, et sans écrire de boucle, écrivez en pseudo-code un algorithme **moyenne** qui prend en entrée un tableau d'entiers *t* et qui renvoie la moyenne des entiers contenus dans *t*.

Q6 (C1b, C1c). Que renvoie **moyenne**([15,9,10,17,8,14]) ?

Exercice 11 – Images en noir et blanc

Le but de ce TD est de manipuler des tableaux de tableaux de booléens qui représentent des images en noir et blanc. En effet, on va coder une image en noir et blanc de la manière suivante : chaque pixel noir correspond au booléen FAUX (pixel éteint) et chaque pixel blanc au booléen VRAI (pixel allumé). Un tableau de booléens correspond donc à une colonne de pixels, et un tableau de tableaux de booléens correspond à une image. Par exemple, l'image ci-contre correspond au tableau [[VRAI,VRAI,VRAI],[VRAI,VRAI,FAUX],[FAUX,FAUX,VRAI]].

Q1 (C1c). Dessinez l'image qui correspond au tableau [[VRAI,VRAI],[FAUX,VRAI],[VRAI,FAUX]].

Q2 (C2a, C2b). Écrivez l'instruction en pseudo-code qui permet de stocker dans une variable *zigzag* le tableau de tableaux de booléens correspondant à l'image suivante :  ?

Q3 (C2a, C2b). Quelle instruction en pseudo-code stocke dans une variable *pixelEnBasADroite* la valeur (VRAI ou FAUX) correspondant à la couleur (blanc ou noir) du pixel en bas à droite de *zigzag* ?

Q4 (C2a, C2b). Écrivez un algorithme **largeurImage** qui prend en entrée un tableau de tableaux de booléens codant une image en noir et blanc et qui renvoie la largeur de l'image correspondante. Par exemple, **largeurImage**(*zigzag*) renvoie 4.

Q4' (C2a, C2b). Même question pour l'algorithme **hauteurImage** (**hauteurImage**(*zigzag*) renvoie 2).

Q5 (C2a, C2b). Écrivez un algorithme **compteBlancsColonne** qui prend en entrée un tableau de booléens et renvoie le nombre de cases contenant VRAI dans ce tableau. Par exemple, **compteBlancsColonne**([VRAI,FAUX,VRAI,VRAI]) renvoie 3.

Q6 (C2a, C2b). Sans faire de boucle (en utilisant un appel de l'algorithme **compteBlancsColonne**), écrivez un algorithme **compteNoirsColonne** qui prend en entrée un tableau de booléens et renvoie son nombre de cases contenant FAUX.

Q7 (C2a, C2b). Écrivez un algorithme **compteBlancs** qui prend en entrée un tableau de tableaux de booléens codant une image et renvoie le nombre de pixels blancs dans l'image codée par ce tableau de tableau de booléens.

Q8 (C2a, C2b). On dispose d'un système qui permet d'afficher un pixel blanc pour 0.2€ par heure, et d'afficher un pixel noir pour 0.1€ par heure. Écrivez un algorithme **coutAffichage** qui prend en entrée un tableau de tableaux de booléens *tabImage* codant une image et deux entiers *h* et *m*, et renvoie le coût d'affichage de l'image pendant *h* heures et *m* minutes.