DS d'algorithme n°1 - Mardi 3 novembre 2015
Durée : 2h - Aucun document autorisé.

Consignes :

- Lisez attentivement le sujet et les rappels de cours avant de commencer.
- Les exercices sont indépendants, vous pouvez les traiter dans l'ordre de votre choix.

Rappels :

- La fonction `print()` sert à afficher un contenu (ce qui se trouve entre les parenthèses) à l'écran.
- On peut construire une chaîne de caractères en concaténant deux (ou plus) chaînes de caractères avec l'opérateur `"+"` ; par exemple `"bonjour" + "monde"` donne la chaîne `"bonjourmonde"`
- Pour construire une chaîne de caractères à partir d'une valeur numérique, on utilise la fonction `str()` ; par exemple `str(42)` donne la chaîne `"42"`.
- On peut aussi construire une chaîne en répétant une autre chaîne un certain nombre de fois, avec l'opérateur `"*"` ; par exemple `"la" * 5` donne `"lalalalala"`.
- On appelle ***spécification formelle***, ou ***contrat***, la déclaration des entrées, sorties, pré-conditions et post-conditions d'un algorithme ou d'une fonction.

Barème

Exercice 1 :	Échauffement	(/2)
Exercice 2 :	Valeur absolue	(/2)
Exercice 3 :	Dessiner des étoiles	(/3)
Exercice 4 :	Mystère (et boule de gomme !)	(/2)
Exercice 5 :	Dessinons (mais pas un mouton) !	(/1)
Exercice 6 :	Des chaînes de caractères	(/2)
Exercice 7 :	Qui est là ?	(/6)
Exercice 8 :	Tables de multiplications	(/3)

Bon courage !

1 Échauffement.....(/2)

- Donnez une définition du terme algorithme.
- Quel opérateur permet de faire une concaténation de chaînes de caractères ?
- Qu'est-ce que le modulo ? Quel opérateur utilise-t-on pour calculer un modulo ?
- Quel est le problème dans la série d'instructions suivante ?

```
a = 42
b = "64"
print(a+b)
```

2 Valeur absolue .....(/2)

Écrire la spécification formelle et l'algorithme `absolue` permettant de calculer la valeur absolue d'un nombre entier passé en paramètre. Par exemple, `absolue(3)` doit retourner `3` et `absolue(-3)` doit retourner `3`.

3 Dessiner des étoiles .....(/3)

Écrire la spécification formelle et l'algorithme `triangleInverse` qui affiche à l'écran un triangle de caractères comme dans les exemples ci-dessous.

```
triangleInverse(5, '*')
*****
****
```

```
***
**
*
```

```
triangleInverse(3, '!')
!!!
!!
```

```
!
```

4 Mystère (et boule de gomme !) .....(/2)

Étudiez l'algorithme suivant, puis supposez que l'on exécute `algoMystere(2,3)`. Recopiez le tableau fourni sur votre copie et complétez-le. Ce tableau présente la trace de l'exécution de l'algorithme. Rappel : la trace de l'exécution d'un algorithme indique les états successifs des variables utilisées dans l'algorithme.

```
def algoMystere(a, b):
    a = a * 2
    b = b + 1
    max = (a*b) // 2
    i = 0
    while i < max:
        c = a + b
        a = a - 1
        b = b + a
        i += 1
```

i	max	a	b	c
-	-	2	3	-
0				

5 Dessinons (mais pas un mouton) ! .....(/1)

Supposons que l'on dispose d'un nouveau langage algorithmique simplifié qui permette de communiquer des instructions à un robot dessinateur. Ce langage contient les instructions suivantes : `avancer(p)` qui permet d'avancer de `p` pas (unité de mesure arbitraire), `tourner(d)` qui permet de tourner de `d` degrés dans le sens horaire, `baisser()` qui indique au robot de baisser le crayon (pour écrire) et `lever()` qui indique au robot de lever le crayon et donc d'arrêter d'écrire. La position initiale du crayon est levée. Mettez vous à la place du robot et dessinez le résultat de l'algorithme pour `M = 3` et `N = 2`.

```
répéter N fois:
        répéter M fois:
             baisser()
             répéter 4 fois:
                  avancer(10)
                  tourner(90)
             lever()
             avancer(20)
        tourner(90)
        avancer(20)
        tourner(90)
        avancer(20)
        tourner(90)
```

6 Des chaînes de caractères .....(/2)

Soit la spécification formelle suivante.

```
'''
    :entree chaine: une chaîne de caractères
    :entree c: un caractère
    :sortie s: un entier
```

```
    :pre-cond: c est un caractère (chaîne de longueur 1)
    :post-cond: s = -1 si la chaîne est vide, sinon, s est égal au nombre
d'apparitions du caractère c dans la chaîne passée en paramètre.
'''
```

- À quel problème répond cette spécification formelle (autrement dit, que fait l'algorithme satisfaisant cette spécification formelle ?).
- Écrire l'algorithme.

7 Qui est là ? ..... (/6)

1. Étant donné la consigne suivante : "*Écrire un algorithme qui permet de vérifier qu'un caractère est bien présent dans une chaîne*", deux personnes proposent les spécifications formelles suivantes. Quels critiques pouvez-vous faire sur ces spécifications ?

```
def algo(c, s):
'''
:entree c: chaîne dans laquelle on
cherche s
:sortie: aucune
:post-cond: on affiche le résultat à
l'écran
'''
```

```
def estPresent(chaine, car):
'''
:entree chaine: une chaîne non vide
:entree car: un caractère
:sortie present: un booleen qui dit si
le caractère est present dans la chaîne
ou non.
'''
```

2. Finalement, voici l'algorithme retenu. Écrivez la spécification formelle correspondante.

```
def indicePresence(chaine, caractere):
    indice = -1
    i = 0
    while i < len(chaine) and indice == -1:
        if chaine[i] == caractere:
            indice = i
        i = i + 1
    return indice
```

3. Écrivez maintenant les spécifications et l'algorithme `estPresentDansChaine` qui parcourt une chaîne de caractère `ch1` et qui affiche "présent" à l'écran si le caractère est présent dans une seconde chaîne `ch2`. Par exemple, `estPresentDansChaine("velo", "hello")` affichera trois fois "présent" car le `e`, le `l` et le `o` de `velo` sont tous présents au moins une fois dans `hello`.

4. Écrivez maintenant un algorithme qui prend en paramètre deux chaînes et qui en fabrique une troisième en effectuant les opérations suivantes. Pour chaque caractère de la première chaîne, on regarde s'il est aussi présent dans la seconde chaîne. Si c'est le cas, on ajoute à la chaîne en cours de création le caractère **suivant** le caractère rencontré dans `ch2`. Sinon, on ajoute à la chaîne le caractère en cours de consultation dans `ch1`.

Exemple : `modifierChaine("velo", "eflm")` retournera "`vfmo`" car :

- `v` n'apparaît pas dans "`eflm`", donc on le recopie tel quel.
- `e` apparaît dans "`eflm`", donc on le remplace par son successeur dans `eflm`, c'est-à-dire `f`.
- `l` apparaît dans "`eflm`", donc on le remplace par son successeur dans `eflm`, c'est-à-dire `m`.
- `o` n'apparaît pas dans "`eflm`", donc on le recopie tel quel.

5. La règle proposée pour construire la chaîne dans la question 4 de cet exercice pose un problème dans un cas. Lequel ?

6. À votre avis, à quoi peut bien servir l'algorithme écrit à la question 4 de cet exercice.

8 Tables de multiplications .....(/3)

1. Écrire un algorithme qui permet d'afficher la table de multiplication (de 0 à 10) d'un entier passé en paramètre.

2. Écrire un algorithme qui permet d'afficher toutes les tables de multiplication des entiers entre 0 et n.