Examen 2022-2023 - CYBER1 (2h00)

Algorithmique - Premiers Pas

NOM : PRÉNOM :

Vous devez respecter les consignes suivantes, sous peine de 0 :
— Lisez le sujet en entier avec attention
— Répondez sur le sujet
— Ne détachez pas les agrafes du sujet
— Écrivez lisiblement vos réponses (si nécessaire en majuscules)
— Vous devez écrire dans le langage algorithmique classique (donc pas de Python ou autre)
— Ne trichez pas

1 Questions (6 points)

1.1 (2 points) Sélectionnez les conditions vraies pour A = 5 et B = 8 :

✓ ((non (A > B)) et (non (B < A))) et ((B != A - 3) ou (A == B - 3))

✓ (non ((A > B) ou (B < A))) et (non ((B == A - 3) ou (A != B - 3)))

☐ (non ((A > B) ou (B > A))) et ((B != A + 3) et (A == B - 3))

✓ (non ((A >= B - 4) et (B >= A + 3))) ou ((B <= A + 3) et (A <= B - 3))

1.2 (2 points) Quelles sont les caractéristiques de cet algorithme :

```
algorithme fonction CalculXYZ : entier
  parametres locaux
    entier    x, y, z
debut
  si (y == 1)
    retourner (1)
  sinon
    si ((x % y) == 0)
      retourne(CalculXYZ(x, (y - 1), z) + y)
    sinon
      retourne(CalculXYZ(x, (y - 1), z))
    fin si
  fin si
fin algorithme fonction CalculXYZ
```

✓ Il est récursif

☐ Il est même récursif terminal

✓ Il s'agit d'une fonction

☐ Il s'agit d'une procédure

1.3 (2 points) Exécutez l'algorithme suivant, et écrivez l'évolution des variables pour $x = 14$ et $y = 5$

```
algorithme fonction CalculXYZ : entier
  parametres locaux
    entier    x, y
  variables
    entier    i, j
debut
i ← 0
j ← 0
tant que (x > 1)
  si ((x % 2) == 1)
    i ← (2 * x) + i
    j ← j - (2 * y)
    y ← (2 * y) + (y / 2)
  sinon
    i ← (2 * x) - i
    j ← 1 - (2 * y)
    y ← y / 2
  fin si
  x ← x / 2
fin tant que
retourner (i + j + x + y)
fin algorithme fonction CalculXYZ
```

x	y	i	j
14	5	0	0
7	2	28	-9
3	5	42	-13
1	12	48	-23
Total	:	38	

2 Algorithmes (14 points)

2.1 (2 points) Écrivez une fonction « *SommeNInt* » itérative calculant la somme des N premiers entiers.

```
algorithme fonction SommeNIntITER : entier
  parametres locaux
    entier    n
  variables
    entier    sum
debut
sum ← 0
tant que (n > 0)
  sum ← sum + n
  n ← n - 1
fin tant que
retourner (sum)
fin algorithme fonction SommeNIntITER
```

2.2 (4 points) Écrivez une fonction « *strlen* » récursive renvoyant la taille d'une chaîne caractères.

```
algorithme fonction strlenRecChapo : entier
  parametres locaux
    char[]    str
debut
retourner (strlenRec(str, 0))
fin algorithme fonction strlenRecChapo

algorithme fonction strlenRec : entier
  parametres locaux
    char[]    str
    entier    len
debut
si (str[len] == '\0')
  retourner (len)
sinon
  retourner (strlenRec(str, len + 1))
fin si
fin algorithme fonction strlenRec
```

2.3 (2 points) Écrivez une fonction « *MedianeTab* » calculant la médiane d'un tableau trié d'entiers.

Pour rappel, la médiane est le nombre au centre d'une distribution triée. Si le tableau a un nombre paire de cases, vous ferez la moyenne des deux éléments centraux.

0	3	9	10	11

La médiane de ce tableau est 9

0	9	10	11

La médiane de ce tableau est $9,5$
$(9 + 10)/2 = 9,5$

```
algorithme fonction medianeTab : entier
  parametres locaux
    entier[]  tab
    entier    len
debut
si (len <= 0)
  retourner (-1)
fin si

si ((len % 2) == 0)
  retourner ((tab[len / 2] + tab[(len / 2) + 1]) /. 2)
sinon
  retourner (len / 2)
fin si
fin algorithme fonction medianeTab
```

2.4 (2 points) Écrivez une procédure « *MoyenneTab* » itérative affichant la moyenne des éléments d'un tableau d'entiers.

```
algorithme procedure MoyenneTabIter : entier
  parametres locaux
    entier[]  tab
    entier    len
  variables
    entier    sum, iter
debut
si (len <= 0)
  ecrire (-1)
sinon
  iter ← len
  sum ← 0
  tant que (iter > 0)
    sum ← sum + tab[iter]
    iter ← iter - 1
  fin tant que
  ecrire(sum /. len)
fin si
fin algorithme procedure MoyenneTabIter
```

2.5 (4 points) Écrivez deux algorithmes « *TabToIntIter* » itératif, et « *TabToIntRec* » en récursif transformant un tableau d'entiers en un unique entier (chaque case contient un nombre positif mais inférieur à 10).

4	0	2	3

Ce tableau doit devenir 4023

```
algorithme procedure TabToIntRecChapo : entier
  parametres locaux
    entier[]  tab
    entier    len
debut
retourner (TabToIntRec(tab, len, 0, 1)
fin algorithme procedure TabToIntRecChapo

algorithme procedure TabToIntRec : entier
  parametres locaux
    entier[]  tab
    entier    len
    entier    acc   # Accumulateur avec valeur finale
    entier    mul   # Multiplicateur conservant la puissance de 10
debut
si (len == 0)
  retourner (acc)
sinon
  retourner (TabToIntRec(tab,
                         len - 1,
                         acc + (mul * tab[len - 1]),
                         mul * 10)
fin si
fin algorithme procedure TabToIntRec
```

```
algorithme procedure TabToIntIter : entier
  parametres locaux
    entier[]  tab
    entier    len
  variables
    entier    total, iter
debut
total ← 0
iter ← 0
tant que (iter < len)
  total ← (total * 10) + tab[iter]
  iter ← iter + 1
fin tant que
retourner (total)
fin algorithme procedure TabToIntIter
```