M2 CCI – Algorithmique – Devoir surveillé

Durée 1h, sans documents **26 octobre 2023**

NE PAS RECOPIER les énoncés des questions. Ne pas perdre de temps à un soin excessif de la présentation. Les questions sont indépendantes. Le barème est indicatif.

Q1 Vérification des types dans une expression algébrique [3 points]

Pour chacune des expressions **E1**, **E2** et **E3** ci-dessous, donner les **contraintes de types** que doivent respecter les noms y apparaissant, puis le **type de l'expression**, dans l'hypothèse où ces contraintes sont respectées (le symbole $\circ$ dénote l'opérateur d'ajout à gauche dans une séquence et l'opérateur $\bullet$ dénote l'ajout à droite dans une séquence).

E1 : si (a et g) alors <b+1,a> sinon <x,g>

E2 : si a+1 > b et y ≠ 'a' alors v sinon y $\circ$ t

E3 : si x = 'y' alors a$\bullet$b sinon "abc"

Q2 Spécification et états [3 points]

Soit l'extrait d'algorithme suivant portant sur **IncrDecr** et **M** :

```
IncDecr : action (donnée-résultat X, Y: entier)
    {IncrDecr(X, Y) : incrémente de 1 la valeur de X et décrémente de 1 la valeur de Y.
    e.i. : X = x0, Y = y0 ; e.f. : X = x0 + 1, Y = y0 - 1}
M (A, B) :
    IncrDecr(A, B)
    A ← A-B
```

— Donner la spécification de l'action **M** (paramètres, états initial et final).

— Quelle est la plus faible pré-condition sur les valeurs initiales **A** et de **B** qu'il faut placer avant l'appel de **IncrDecr** pour que la valeur finale de **A** soit toujours strictement positive à l'issue de l'exécution de **M** ?

Q3 Dénombrement d'opérations [3 points]

On étudie l'algorithme suivant :

```
(1)    x ← 1
(2)    pour i allant de 1 a n
(3)        x ← x *i
(4)        pour k allant de 1 a n
(5)            pour j allant de 1 a k
(6)                x ← x * 2
```

— **Donner le nombre d'applications de l'opérateur *** engendré par une exécution de l'algorithme. Justifier la réponse fournie par un raisonnement sur l'algorithme. On pourra pour cela se référer aux numéros des lignes de l'algorithme.

Q4 À propos d'une compétition sportive [5 points]

On représente les résultats d'une compétition sportive comportant 3 épreuves par un ensemble de *scores* : un *score* est un quadruplet, associant, à chaque nom de sportif participant à la compétition, la note (sur 10) qu'il a obtenue à chacune des 3 épreuves, par exemple <"durand", 5, 7, 3>.

Un ensemble des scores est une séquence fournie dans un tableau avec longueur explicite. Si la séquence de scores est représentée dans un tableau **T**, **T**$_{\mathbf{i}}$ est le score du sportif de numéro **i**.

Ceci est formalisé par le lexique suivant :

Lmax : constante de type entier > 0 Score : type <Nom : texte ; Notes : tableau sur [1…3] d'entier sur [0…10]>

On spécifie la fonction suivante :

ExisteChampion : fonction (T : tableau sur [1…Lmax] de Score, L : entier sur [0…Lmax])
→ booléen, texte

{soit <B,N> = ExisteChampion(T, L). B a la valeur vrai, si, dans la séquence représentée dans T et ayant la longueur L, il existe un score avec la note maximale 10 à toutes les épreuves. N est alors le nom du premier sportif correspondant. Si B a la valeur faux, la valeur de N n'est pas significative.}

— Donner une **réalisation** de la fonction **ExisteChampion.**

Q5 Nombre de valeurs distinctes d'une séquence [6 points]

On s'intéresse aux valeurs différentes d'une séquence d'entiers. Par exemple :

dans la séquence [**3**, 3, **-1**, 3, **7**, **0**,7, **6**, 3, 7, -1, **-15**, 3],
il y a 6 valeurs différentes : [**3**, **-1**, **7**, **0**, **6**, **-15**].

Les séquences sont représentées dans des tableaux avec longueur explicite.
On spécifie une fonction nommée **NbDif** :

Lmax : constante de type entier > 0

NbDif : (T : tableau sur [1…Lmax] d'entier, L : entier sur [0…Lmax]) → entier ≥ 0

{ Nombre de valeurs différentes dans la séquence représentée dans le tableau T et de longueur explicite L. }

Pour réaliser cette fonction, on propose le principe suivant : « dans un parcours de la séquence, augmenter un compteur de 1, chaque fois qu'une valeur apparaît pour la **première fois** »; une valeur apparaît pour la première fois, si elle ne fait pas partie des éléments qui la précèdent dans la séquence.

— Donner une **réalisation** de la fonction **NbDif** selon le principe ci-dessus.

Q6 Question subsidiaire : Invariants d'itérations (bonus jusqu'à 3 points)

On considère l'algorithme suivant :

```
a : entier ≥ 0
u, v : entier ≥ 0
u ← x ; v ← y ; a ← 0
{*P1*}
tant que v ≠ 0
        {*P2*}
        si v reste 2 ≠ 0 alors {v est impair} a ← a + u
         u ← 2*u ; v ← v quotient 2
        {*P3*}
{*P4*}
```

— Démontrer que l'assertion "u*v + a = x *y" est un invariant de l'itération.

— En déduire le résultat de la fonction, et justifier la réponse.

M2 CCI – Algorithmique

AL – Quick 1 : des exemples de solutions **26 octobre 2023**

Q1 Vérification des types dans une expression algébrique [3 points, 1 point par expression]

E1 : si (a et g) alors <b+1,a> sinon <x,g>
E2 : si a+1 > b et y ≠ ‘a’ alors v sinon $y \circ t$
E3 : si x = ‘y’ alors a•b sinon ″abc″

E1 : **a** et **g** sont de type booléen, **b** et **x** sont de type entier. **E1** est de type <entier,booléen>.
E2 : **a** et **b** sont de type entier, **y** de type caractère, **t** et **v** de type texte. **E2** est de type texte.
E3 : **x** et **b** sont de type caractère, **a** de type texte. **E3** est de type texte.

Q2 Spécification et états [3 points]

M : action (donnée-résultat A, B : entier)
{e.i. : $A = a_0$, $B= b_0$; e.f. : $A = a_0-b_0+2$, $B= b_0-1$}

Pour que la post-condition $\boldsymbol{a_0-b_0+2 > 0}$, il faut que $\boldsymbol{a_0 > b_0 -2}$.

Q3 Dénombrement d'opérations [3 points]

— ligne 3 : une exécution de la ligne 3 engendre une multiplication.
— lignes 4 à 6 : une exécution de la ligne 6 engendre une multiplication ; par conséquent, une exécution des lignes 5-6 engendre **k** multiplications ; par conséquent, une exécution des lignes 4 à 6 engendre $\sum_{k=1 \text{ à } n} k = n(n+1)/2$ multiplications.
— Des deux points précédents, on déduit que pour **n** donné, une exécution des lignes 3 à 6 engendre **n(n+1)/2 + 1** multiplications.
— pour **n** donné, l'exécution de l'algorithme engendre donc $\sum_{i=1 \text{ à } n} (n(n+1)/2 + 1)$ multiplications, c'est-à-dire : $n^2(n+1)/2 + n$.

Q4 À propos d'une compétition sportive [5 points]

Réalisation version 1 : application du schéma de parcours

ExisteChampion(T, L) :
  Existe : booléen
  N : texte
  Existe ← faux
  N ← []
  pour i allant de 1 à L
    {Existe a la valeur vrai si et seulement il existe un champion dans $T_{[1\ldots i-1]}$}
    Si non Existe et $(T_i.Notes)_1 + (T_i.Notes)_2 + (T_i.Notes)_3 = 30$ alors
      N ← T_i.Nom
      Existe ← vrai
 retour : <Existe, N>

Réalisation version 2 : application du schéma de recherche avec arrêt dès que possible

ExisteChampion(T, L) :
  i : entier sur [1…Lmax+1]
  i ← 1
  tant que i ≤ L et puis $((T_i.Notes)_1 + (T_i.Notes)_2 + (T_i.Notes)_3 \neq 30)$
    i ← i + 1
 retour : si i ≤ L alors <vrai, T_i.Nom> sinon <faux, []>

Q5 Nombre de valeurs distinctes [6 points]

Pour vérifier, lors du parcours de la séquence, qu'un élément apparaît pour la première fois, on peut appliquer un schéma de parcours ou un schéma de recherche.

{version 1 : parcours}

```
NbDif(T, L) :
    C : entier ≥ 0                               {compteur : pour élaborer le résultat}
    Existe : booléen                             {pour vérifier l'existence}
    C ← 0
    pour i allant de 1 a L
        {C est le nombre de valeurs distinctes déjà rencontrées}
        Existe ← faux
        pour j allant de 1 a i-1
            Existe ← Existe ou Ti = Tj
        si non Existe alors C ← C + 1            {Ti apparaît pour la première fois}
retour : C
```

{version 2 : recherche – avec technique de sentinelle}

```
NbDif(T, L) :
    C : entier ≥ 0                               {compteur : pour élaborer le résultat}
    j : entier sur [1…Lmax]                      {pour vérifier l'existence}
    C ← 0
    pour i allant de 1 a L
        {C est le nombre de valeurs distinctes déjà rencontrées}
        {schéma de recherche. Ti sert de sentinelle.}
        j ← 1
        tant que Tj ≠ Ti   { ou :  tant que j ≠ L+1 et puis Tj ≠ Ti }
            j ← j + 1
        si j = i alors C ← C + 1                 {Ti apparaît pour la première fois}
retour : C
```

Q6 Subsidiaire : À propos d'invariants d'itérations (3 points max)

Supposons : *{u0*v0+ a0 = x*y}* en **P2** avec u0, v0 et a0 les valeurs de u, v et a en **P2**

```
    si v reste 2 ≠ 0 alors {il existe k tel que : v0 = 2*k+1}
        a ← a + u   {a1 = a0 + u0}
        u ← 2*u ; v ← v quotient 2
        {u1=2u0, v1=k.
        Donc : u1*v1 + a1=2u0*k + a0 + u0 = u0*(2k+1)+a0 = u0*v0+a0 = x*y}
    sinon {il existe k tel que : v0 = 2*k}
        u ← 2*u ; v ← v quotient 2
        {a1=a0, u1=2u0, v1=k.
        Donc : u1*v1 + a1=2u0*k + a0 = u0*v0+a0 = x*y}
```

Donc, en **P3**, dans les deux cas, on a : { u1*v1+ a1= x*y}

Donc *{u*v + a= x*y}* est un invariant

En **P1**, x*y+0 = x*y donc l'invariant est vrai aussi en **P1**.

L'itération termine car v est un entier positif qui ne fait que diminuer par quotient de 2.
A l'issue de l'itération (**P4**), l'invariant est vrai et on a v = 0, donc u*0 + a = x*y.
On en déduit a = x*y.