M2 CCI – Algorithmique – Devoir surveillé

Durée 1h, sans documents **23 octobre 2025**

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 & dénote l'opérateur de concaténation sur les textes).

E1 : si a alors 25 sinon b
E2 : < y=3.5, "ab" & a, x+y >
E3 : (a+1 > b) ou non x

Q2 Statut des paramètres d'une action [3 points]

— **Compléter** (• • • •) les spécifications suivantes (type et statut des paramètres) :

> CalculerQuotient : action (• • • •)
> *{CalculerQuotient(X, Y, Z) calcule dans Z le quotient de la division entière de X par Y, tous deux positifs}*
> Accumuler : action (• • • •)
> *{Accumuler(X, Y) : e.i. X = x0 et Y = y0 ; e.f. : X = x0 + y0 et Y = y0 }*

Q3 Suppression [8 points]

On considère une séquence d'entiers donnée sous forme contiguë dans un tableau **T** avec longueur explicite **L**.

Le lexique correspondant est le suivant :

Lmax : constante de type entier > 0
T : tableau sur [1…Lmax] d'entier
L : entier sur [0…Lmax]
X : entier

(i) Suppression du premier élément de valeur X

— **Donner la réalisation** d'un algorithme de **suppression du premier élément** de valeur **X** (dont la valeur est supposée connue) dans la séquence de longueur **L** représentée dans le tableau **T** (dont les valeurs sont aussi supposées connues). **L'ordre des éléments dans la séquence doit être conservé**. Si **X** n'existe pas dans la séquence, la séquence est inchangée.

Exemple :
Si les données sont **:** $\mathbf{T}_{[1...L]}$ = [6 ;4 ;7 ;**3** ;7 ;3 ;5 ;7]**,** **L** = 8, **X**=3 alors le résultat attendu est $\mathbf{T}_{[1...L]}$ = [6 ;4 ;7 ;7 ;3 ;5 ;7]**,** **L** = 7.

(ii) Suppression de tous les éléments de valeur X

— **Donner la réalisation** d'un algorithme de **suppression de tous les éléments** de valeur **X** (dont la valeur est supposée connue) dans la séquence de longueur **L** représentée dans le tableau **T** (dont les valeurs sont aussi supposées connues). **L'ordre des éléments dans la séquence doit être conservé**. Si **X** n'existe pas dans la séquence, la séquence est inchangée.

Exemple :
Si les données sont **:** $\mathbf{T}_{[1...L]}$ = [6 ;4 ;7 ;**3** ;7 ;**3** ;5 ;7]**,** **L** = 8, **X**=3 alors le résultat attendu est $\mathbf{T}_{[1...L]}$ = [6 ;4 ;7 ;7 ,5 ;7]**,** **L** = 6.

Q4 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.

Q5 Question subsidiaire : Invariants d'itérations (bonus jusqu'à 2 points)

L'algorithme suivant traite un tableau d'entiers **T** défini sur l'intervalle **[1…n]** ($n > 0$) :

```
i ← 1 ; j ← n
tant que i ≠ j
        si Tᵢ ≤ Tⱼ alors i ← i + 1
        sinon j ← j - 1
Écrire(Tᵢ)
```

— Pour chacune des trois assertions ***A1***, ***A2*** et ***A3*** suivantes, indiquer si c'est un invariant de l'itération. Justifier la réponse.

A1 : $T_1 < T_n$

A2 : $j > 0$

A3 : *la valeur maximum de T se trouve dans* $T_{[i...j]}$

Remarque : On rappelle qu'une propriété est un invariant dès qu'elle est préservée par l'exécution d'une itération.

— Démontrer que l'algorithme affiche la valeur maximum des éléments du tableau **T**.

M2 CCI – Algorithmique

AL – DS 1 : des exemples de solutions **23 octobre 2025**

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

— **E1** : **a** est de type booléen, **b** de type entier. **E1** est de type entier.
— **E2** : **a** est de type texte, **x** et **y** de type réel. **E2** est de type <booléen, texte, réel>.
— **E3** : **a** et **b** sont de type entier, **x** de type booléen. **E3** est de type booléen

Q2 Statut des paramètres d'une action [3 points]

CalculerQuotient : action (donnée X : entier ≥ 0, Y : entier > 0 ;
résultat Z : entier ≥ 0)
Accumuler : action (donnée-résultat X : entier ; donnée Y : entier)
{Remarque : on peut aussi spécifier les paramètres comme étant des réels.}

Q3 Suppression [8 points]

(i) Suppression du premier élément de valeur X [4 points]

```
i ← 1                                              {recherche de X}
tant que i ≠ L+1 et puis Tᵢ ≠ X
        i ← i + 1
si i<=L alors
  pour k allant de i a L-1
     Tk ← Tk+1
  L ← L – 1
```

(ii) Suppression de tous les éléments de valeur X [4 points]

```
i ← 1
tant que i ≠ L+1
    si Ti = X alors
      pour k allant de i a L-1
         Tk ← Tk+1
      L ← L – 1
    sinon
      i ← i + 1
```

Q4 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 ≠ i et puis Tj ≠ Ti }
            j ← j + 1
        si j = i alors C ← C + 1           {Ti apparaît pour la première fois}
retour : C
```

Q5 Subsidiaire : À propos d'invariants d'itérations (2 points max)

— L'assertion ***A1*** *"$T_1 < T_n$"* est un invariant de l'itération : le corps de l'itération ne modifie ni la valeur de **n** ni la valeur de **T** : l'hypothèse ***A1*** en P2 implique ***A1*** en P3.

— L'assertion **A2** *"$j > 0$"* n'est pas un invariant de l'itération : supposons qu'en P2, **i= 3**, **j = 1**, **T_1=5** et **T_3=7** ; alors en P3, après l'exécution du corps de l'itération, j vaut 0 ce qui contredit ***A2***.

— L'assertion ***A3*** est un invariant. Hypothèse : ***A3*** est satisfaite en P2. Soient **i0** et **j0** les valeurs de **i** et **j** en ***P2***. On examine les deux cas de la conditionnelle :

- **$T_{i0} \le T_{j0}$** : en P3, **j = j0** et **i = i0+1** ; la valeur **T_{i0}** n'est plus dans $T_{[i0+1\ldots j0]}$.
  - si **$T_{i0} < T_{j0}$**, **T_{i0}** n'a pas la valeur maximum de **T**, ce qui assure que ***A4*** est satisfaite.
  - si **$T_{i0} = T_{j0}$**, ***A4*** est satisfaite car **T_{j0}** reste dans l'ensemble.
- **$T_{i0} > T_{j0}$** : l'invariance de ***A*** est montrée par un argument similaire au précédent, car **T_{j0}** n'a certainement pas la valeur maximum de **T**.

— L'assertion ***A3*** est un invariant. Elle est vraie en **P1** car il est vrai que le maximum de **T** se trouve dans **$T_{[1..n]}$**. Le programme termine car **i** croit et **j** décroit donc **i=j** devient vrai à un instant donné.

Donc en **P4**, la valeur maximum de **T** se trouve dans **$T_{[i..j]}$** et **i=j**. Donc la valeur maximum de **T** est dans **T_i**.