

**Consignes**

- Le seul document autorisé est une feuille A4 manuscrite recto-verso.
- Tous les appareils électroniques doivent être éteints et rangés hors de vue.
- Vous devez rendre le sujet complété avec votre copie. Assurez-vous d'y avoir apposé votre nom.
- Sauf mention explicite du contraire, vos programmes Heptagon doivent être bien typés, causaux, bien initialisés et utiliser les horloges de façon cohérente.

**1 Matrices**

Dans cet exercice, on va implémenter une petite bibliothèque de manipulation de matrices  $2 \times 2$  à coefficients entiers. Ces matrices sont définies par les types **type** `vec2 = int^2` et **type** `mat4 = vec2^2`. Une valeur `[[a, b], [c, d]]` de type `mat4` représente la matrice ci-contre. Autrement dit, une matrice est ici implémentée par une paire de vecteurs lignes, ceux-ci étant des paires d'entiers.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

1. Définir une fonction

```
fun vv(x, y : vec2) returns (o : int)
```

telle que le  $n$ ème entier de la suite `vv(x, y)` soit le produit scalaire des  $n$ èmes vecteurs des suites `x` et `y`. (On rappelle que le produit scalaire de deux vecteurs est donné la somme des produits des coefficients de même rang; par exemple, le produit scalaire de `[1, 2]` et `[3, 2]` est 7.)

2. Définir deux fonctions

```
fun line(x : mat4, i : int) returns (l : vec2)
fun column(x : mat4, j : int) returns (c : vec2)
```

telles que les  $n$ ème valeurs des flots `line(x, i)` et `column(x, j)` soient respectivement la  $i$ ème ligne et la  $j$ ème colonne de la  $n$ ème valeur du flot `x`.

3. Définir en utilisant les fonctions précédentes une fonction

```
fun mm(x, y : mat4) returns (z : mat4)
```

qui reçoit deux suites de matrices `x` et `y` et calcule la suite de leurs produits, c'est-à-dire la suite de matrices  $z$  telle que  $z_n = x_n \cdot y_n$ , où  $\cdot$  désigne le produit de matrices. Par exemple, on doit avoir

$$\text{mm}([ [1, 0], [2, 3] ], [ [4, 1], [5, 3] ]) = [ [4, 1], [23, 11] ].$$

4. Soient `m : mat4` et `u : int` les suites définies par les équations suivantes.

```
m = [ [1, 0], [0, 1] ] fby mm([ [0, 1], [1, 1] ], m);
u = vv(line(m, 0), [1, 1]);
```

Remplir le chronogramme ci-dessous *directement sur le sujet*.

m						
u						

5. La suite `u` définie à la question précédente est bien connue. Donner son nom.

## 2 Compilation des automates

Le code ci-dessous a été produit par la passe d'élimination des automates du compilateur Heptagon.

```
type st = St_Idle | St_Count

node f_trad(m : bool; e : bool; r : bool) returns (o : int)
var s, ns : st; nr, r_1, pnr : bool;
let
  switch (St_Idle fby ns)
  | St_Idle
    do reset (s, r_1) = if m then (St_Count, false) else (St_Idle, pnr)
    every pnr
  | St_Count
    do reset (s, r_1) = if r then (St_Count, true)
                        else if m then (St_Idle, true)
                        else (St_Count, pnr)
    every pnr
  end;
switch (s)
  | St_Idle
    do reset (ns, nr) = (St_Idle, false); o = 0
    every r_1
  | St_Count
    do reset (ns, nr) = (St_Count, false); o = 0 fby (o + if e then 1 else 0)
    every r_1
  end;
  pnr = false fby nr
tel
```

1. Le code ci-dessus est produit par un schéma de traduction général, et contient donc des redondances. Donner une version simplifiée au maximum. En particulier, vous éliminerez les variables qui définissent des suites constantes et les réinitialisations inutiles. Vous décrirez en quelques mots les simplifications effectuées.
2. Donner le code source de l'automate dont le code ci-dessus est la traduction. Votre proposition doit être la plus simple possible, et reposer sur le schéma d'élimination des automates vu en cours.

## 3 Moyenne glissante

1. Définir une fonction

```
fun shiftr<<n : int>>(first : int; x : intn) returns (o : intn)
```

telle que le  $i$ ème élément de la sortie  $o$  est le  $i$ ème élément de  $x$  décalé d'une cellule vers la droite, avec la  $i$ ème valeur de  $first$  comme premier élément. Par exemple, `shiftr<<3>>(42, [1, 2, 3])` doit calculer la suite constante `[42, 1, 2]`.

2. Définir un nœud

```
node window<<n : int>>(ini : int; x : int) returns (o : intn)
```

tel que le  $i$ ème élément de la sortie  $o$  est un tableau de taille  $n > 0$  dont la  $j$ ème cellule est le  $(i - j)$ ème élément du flot  $x$ , ou bien  $ini_0$  si  $i - j < 0$ . En particulier, si  $i \geq n - 1$ , alors le tableau  $o_i$  contient les  $n$  dernières valeurs de  $x$ , la plus récente d'abord :  $o_i = [x_i, x_{i-1}, \dots, x_{i-n+1}]$ .

3. Définir un nœud

```
node average<<n : int>>(x : int) returns (o : int)
```

tel que le  $i$ ème élément de la sortie  $o$  est la moyenne des  $n$  dernières valeurs du flot  $x$  si  $i \geq n$ . Vous pouvez fixer librement la valeur de la sortie lorsque  $i < n$ .