Université P. & M. Curie
Licence Mention Informatique – L3
Programmation Concurrente (LI330)

Examen Final
17 Juin 2013
Durée : 2 heures

Les téléphones portables doivent être éteints et rangés dans vos sacs.
Le mémento Ada/C est le seul document autorisé pendant l'épreuve.

Le barème (sur 20) est donné à titre indicatif.
Il vous est conseillé de soigner votre copie et de rédiger des réponses claires (en particulier les programmes).
Toutes vos réponses doivent être dûment justifiées.
Lisez attentivement le sujet. Toute réponse hors sujet sera considérée comme fausse.
L'oubli des **`.all`** *(à bon escient) dans la gestion des pointeurs sera considéré comme une erreur.*

Exercice 1 – Tunnel à à une voie (12 points)

Nous considérons un tunnel qui ne comporte qu'une seule voie mais dans lequel des trains circulent dans les deux sens. Le tunnel considéré doit donc être partagé par les trains qui le traversent. Pour cela, nous devons développer un contrôleur suivant le schéma indiqué en figure 1.

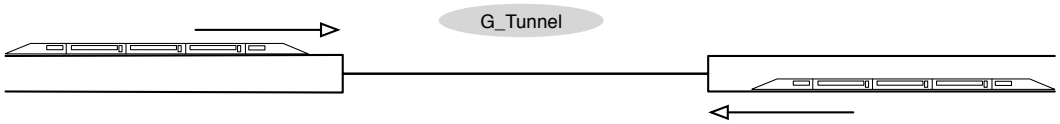

FIGURE 1 – Architecture du tunnel partagé

Il existe deux types de trains : allant de `Gauche` à `Droite` et de `Droite` à `Gauche`. Ces trains sont tous basés sur un modèle unique de `Train` qui intègre les deux comportements.

Un gestionnaire `G_Tunnel` gère les accès au tunnel. Il se contente d'attendre les requêtes (potentiellement bloquantes) des trains qui souhaitent entrer dans le tunnel. Lorsque `Alternance` trains sont passées dans une direction, le changement de sens est effectué.

On dispose dans un premier temps des déclarations suivantes :

```
-- Constantes
Alternance : constant Positive := 3;

-- Type pour designer les destinations
type Destination is (gauche, Droite);

-- Une tache Train
type Train;
type A_Train is access Train;
task type Train (Id : Integer ; Dest : Destination);
```

Question 1 – 0,5 point

Justifiez le choix d'implémenter le gestionnaire du tunnel `G_Tunnel` au moyen d'un objet protégé.

Pour passer le tunnel, un `Train` doit demander l'autorisation au `G_Tunnel` (cette autorisation est bloquante). Les autorisations sont différenciées en fonction de la direction du train. Chaque `Train` signale au `G_Tunnel` qu'il sort du tunnel (cette requête n'est jamais bloquante). Lors d'une demande d'entrée (quelque soit le sens) et lorsqu'il sort, un `Train` transmet son identité.

Pour assurer le bon fonctionnement du tunnel, `G_Tunnel` stocke trois données : le nombre de trains qui sont entrés depuis le changement de direction (`Entres`), le nombre de trains qui sont sortis depuis le changement de direction (`Sortis`) et la direction courante (`Sens`).

Question 2 – 1 point

Écrivez le source de la déclaration de `G_Tunnel`.

Question 3 – 1,5 points

Écrivez le source du corps d'une tâche `Train`.

Question 4 – 1 point

Déterminez la condition sur les variables de `G_Tunnel` qui permet à un `Train` allant de Gauche à Droite de passer. Justifiez brièvement votre réponse.

On s'intéresse maintenant à la condition qui permet de changer de direction, quel que soit le trajet du train.

Question 5 – 1 point

Déterminez la condition sur les variables de `G_Tunnel` qui permet de changer le sens de passage dans le tunnel. Justifiez brièvement votre réponse.

Question 6 – 1 point

Au niveau de quel service testerez-vous la condition de changement ? Justifiez brièvement votre réponse.

Question 7 – 3,5 points

Programmez le corps de `G_Tunnel`.
Nous définissons, pour la suite de ce problème, la constante suivante :

```
   Max_Train    : constant Positive := 4 * Alternance;
```

Question 8 – 2,5 points

Écrivez le source du programme principal qui initialise deux tableaux pour les `Trains` (nommés `Les_gauche_Droite` et `Les_Droite_gauche`). Ces tableaux sont dimensionnés avec la constante `Max_Train`. Tous les `Train` sont créés avec des identificateurs différents.

Exercice 2 – Tâches & objets protégés (8 points)

Considérons le programme suivant :

```
with Ada.Text_Io, Ada.Numerics.Discrete_Random;
use Ada.Text_Io;

procedure Tasks is
   --| Pour la generation de durees aleatoires
   package Alea_Int is new Ada.Numerics.Discrete_Random (Integer);
   Graine : Alea_Int.Generator;
   function Calcule_Attente return Duration is
   begin
      return Duration (Alea_Int.Random (Graine) mod 10) * 0.1;
   end Calcule_Attente;

   --| Tache Serveur
   task Serveur is
      entry Service_1 (Who : in Natural);
      entry Service_2 (Who : in Natural);
      entry Service_3 (Val : out Integer);
   end Serveur;

   task body Serveur is
      Valeur : Integer := 0;
   begin
      loop
         select
            when Valeur /= 0 =>
            accept Service_1 (Who : in Natural) do Valeur := Valeur + Who;
            end Service_1;
         or
            accept Service_2 (Who : in Natural) do Valeur := Valeur - Who;
            end Service_2;
         or
```

```
32            accept Service_3 (Val : out Integer) do Val := Valeur;
33            end Service_3;
34         end select;
35         delay Calcule_Attente;
36      end loop;
37   end Serveur;
38
39   --| Tâche client
40   task type Client (Id : Natural);
41   type A_Client is access Client;
42
43   task body Client is
44
45   begin
46      for I in 1.. 5 loop
47         if Id mod 2 = 0 then
48            Serveur.Service_1 (Id);
49         else
50            Serveur.Service_2 (Id);
51         end if;
52         delay Calcule_Attente;
53      end loop;
54      Put_Line ("Le client " & Natural'Image (Id) & " se termine.");
55   end Client;
56
57   Les_Clients : array (1 .. 5) of A_Client;
58   V           : Integer;
59
60   --| Programme Principal
61 begin
62   Alea_Int.Reset (Graine);
63   for I in Les_Clients'range loop
64      Les_Clients(I) := new Client (I);
65   end loop;
66   for I in 1..20 loop
67      Serveur.Service_3 (V);
68      Put_Line ("Je dispose de la valeur: " & Integer'Image (V));
69      delay Calcule_Attente;
70   end loop;
71 end Tasks;
```

Question 1 – 0,5 point

Combien de tâches sont-elles créées par ce programme. Justifiez brièvement votre réponse.

Question 2 – 0,5 point

Expliquez pourquoi ce programme assure la cohérence de la variable `Valeur` ?

Question 3 – 1 point

Pourquoi ce programme ne se termine-t-il pas ?

Question 4 – 1 point

Que faut-il ajouter pour que ce programme se termine ?

Question 5 – 1,5 points

Expliquez pourquoi on peut transformer la tâche `serveur` en un objet protégé.

Question 6 – 1,5 points

Reprogrammez (spécification et corps) la tâche `serveur` sous la forme d'un objet protégé. ***Nota Bene :*** *Vous justifierez la conversion de chacun des trois points d'entrée de la tâche* `Serveur`.

Question 7 – 2 points

Représentez sous la forme d'un chronogramme l'exécution de ce programme lorsqu'il y a une unique tâche client. On considère qu'il n'y a commutation que lorsqu'une tâche est bloquée. Vous ne tiendrez pas compte des instructions **`delay`**. Vous indiquerez à quelle ligne de programme correspondent les changements d'états .