Université P. & M. Curie Examen Final
Licence Mention Informatique – L3 18 Janvier 2012
Programmation Concurrente (LI330) 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 – Hangar de stockage de locomotives (16 points)

On considère un ensemble de hangars composé de N unités pouvant chacune accueillir une locomotive. Le segment d'accueil permet d'entrer et de sortir des locomotives de la zone de stockage. Une locomotive entrante est dirigée vers un hangar donné (ou vers le segment d'accueil si elle est sortante) au moyen d'un segment de voie tournant. L'architecture du système est indiquée en figure 1.

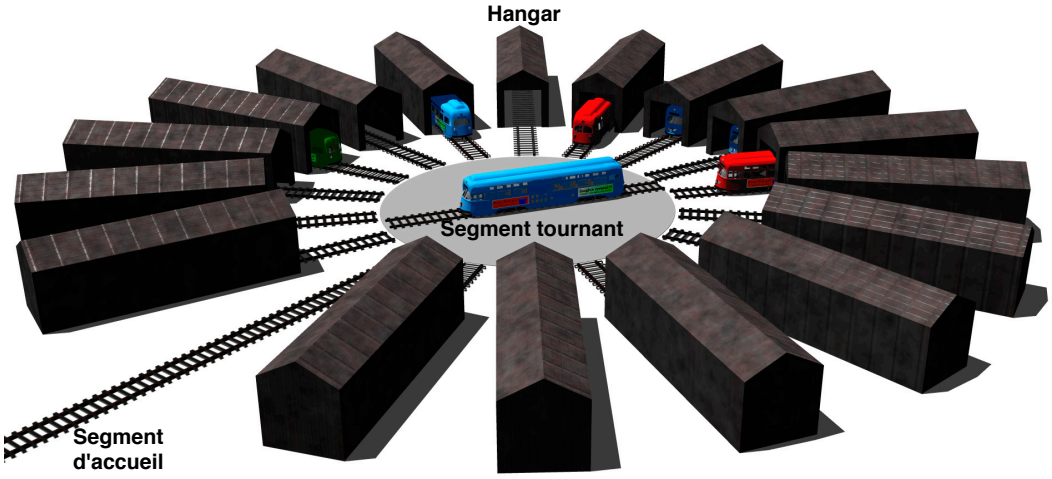

FIGURE 1 – Architecture du système pour $N = 17$

Question 1 – 1 point

Indiquez, en le justifiant, s'il existe des ressources critiques dans le système. Si oui, précisez lesquelles.

Question 2 – 1 point

Indiquez, en le justifiant, s'il existe des ressources banalisées dans le système. Si oui, précisez lesquelles.

On souhaite représenter au moyen d'une application concurrente le pool de hangars ainsi que les locomotives qui s'y garent. Pour cela, on considèrera un système composé des éléments suivants :

- des locomotives,
- des hangars,
- le segment tournant.
- le segment d'entrée.

Une *locomotive* entre dans le pool de hangars (en passant par le segment d'accueil), appelle le segment tournant qui se positionne devant le segment d'accueil, entre dans le segment tournant puis le positionne devant le hangar identifié par

un numéro, s'y gare un moment, ressort du hangar vers le segment tournant (en ayant réservé auparavant le segment tournant pour une sortie) puis quitte le pool de hangars en passant par le segment d'accueil (qui doit être libre).

Un *hangar* : une locomotive peut entrer dans un hangar s'il et vide et peut en sortir s'il est plein.

Le *segment tournant* peut accueillir une locomotive s'il est vide, se positionne devant le hangar (ou le segment d'accueil) selon ce qui est spécifié par la locomotive. On considère que les positions licites du segment tournant sont numérotées de 0 à N. La position 0 représente le segment d'accueil.

Le *segment d'accueil* peut accueillir une locomotive qui rentre ou qui sort du système.

On considère pour l'instant les hypothèses suivantes : le système est composé de N hangars et n'accueille pas plus de N locomotives. On supposera qu'une locomotive numérotée i doit être garée dans le hangar i.

Le système est décrit par les structures indiquées ci-dessous où l'on considère que le système comporte N locomotives et N hangars (ici, $N = 5$) :

```
1   -- Constantes pour dimentionner le pool de hangars
2   Max_Hangar : constant Natural := 5;
3   Max_loco : constant Natural := 5;
4
5   -- types requis
6   type Id_Loco is range 1 .. Max_loco;
7   type Position is range 0 .. Max_Hangar;
8   subtype Id_Hangar is Position range 1 .. Position'Last;
9
10  -- les interlocuteurs du systeme (specification)
11  task type Locomotive (My_Id  : Id_Loco;
12                        Hangar : Id_Hangar) is
13      entry Demarrer;
14  end Locomotive;
15
16  protected type Hangar (My_Id : Id_Hangar) is
17      -- Entrer dans le hangar (si c'est possible)
18      entry entrer (Id : in Id_Loco);
19      -- Sortie du hangar (si c'est possible)
20      entry sortir;
21  private
22      Plein        : Boolean := False;
23      Id_Si_Plein : Id_Loco;
24  end Hangar;
25
26  task Segment_Accueil is
27      -- Entrer et sortir
28      entry Entrer (Id : in Id_Loco);
29      entry Sortir;
30      -- Reserver le segment pour une sortie
31      entry Reserver_Sortie_PA (Id : in Id_Loco);
32      -- Entrer dans le segment qui a ete reserve
33      entry Entrer_Reserve (Id : in Id_Loco);
34  end Segment_Accueil;
35
36  task Segment_Tournant is
37      -- Appeler le segment tournant a une position donnee
38      entry Appeler (Ou  : in Position;
39                     Qui : in Id_Loco);
40      -- Entrer dans le segment tournant
41      entry Entrer (Id : in Id_Loco);
42      -- Positionner le segment tournant a l'endroit ou on veut sortir
43      entry Se_Positionner (Ou  : in Position);
44      -- Sortie
45      entry Sortir;
46  end Segment_Tournant;
47
48  -- Les donnees representant les differentes taches
49  type A_Locomotive is access Locomotive;
50  Les_Locomotives : array (Id_Loco) of A_Locomotive;
51  type A_hangar is access Hangar;
52  Pool_Hangars : array (Id_Hangar) of A_hangar;
```

Question 3 – 0,5 point

Combien de tâches le système aura-t-il vu à l'issue de l'exécution du programme

Question 4 – 1 point
Écrivez le source du programme principal qui initialise les structures de données `Pool_Hangars` et `Les_Locomotives` avant de démarrer les locomotives.

Question 5 – 1,5 point
Expliquez pourquoi une locomotive qui sort de son hangar doit réserver le segment d'accueil avant de commencer tout mouvement.

Question 6 – 1 point
Programmer le corps de la tâche `Locomotive`.

Question 7 – 2 point
Programmer le corps du type protégé `Hangar`.

Question 8 – 1 point
Transformez le type protégé `Hangar` en une tâche.

Question 9 – 2,5 points
Programmer le corps de la tâche `Segment_Tournant`.

Question 10 – 2,5 points
Programmer le corps de la tâche `Segment_Accueil`.

Question 11 – 2 points
Quelle que soit la valeur de N, à quelle condition le système peut-il fonctionner lorsque les locomotives 1 et 2 souhaitent se garer dans le hangar 1 ? justifiez brièvement votre réponse.

Exercice 2 – Questions de cours (4 points)

Question 1 – 2 points
Énoncez les quatre conditions qui rendent l'interblocage possible dans un système.
Considérons le programme ci-dessous :

```
procedure Chrono is

   task type T1 is
      entry E;
   end T1;

   task type T2;

   task body T1 is

   begin
      Put_Line ("coucou");
      accept E;
   end T1;

   task body T2 is
   begin
      Put_Line ("hello");
   end T2;

   type A_T1 is access T1;
   P1, P2 : A_T1;

begin
   P1 := new T1;
   P2 := new T1;
```

```
   P1.all.E;
   declare
      A1 : T2;
   begin
      null;
   end;
   declare
      A2 : T2;
   begin
      null;
   end;
   P2.all.E;
end Chrono;
```

Question 2 – 2 points

Construisez le chronogramme de ce programme