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

Examen partiel
12 Novembre 2009
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.

Exercice 1 – Problème : gestion d'une écluse, 12 points

On se propose de développer un système de gestion pour une écluse construite suivant le schéma de la Figure 1. Le système est composé :

- de deux canaux sur lesquels circulent des péniches (un canal par sens),
- d'une écluse unique permettant de monter et descendre les péniches.

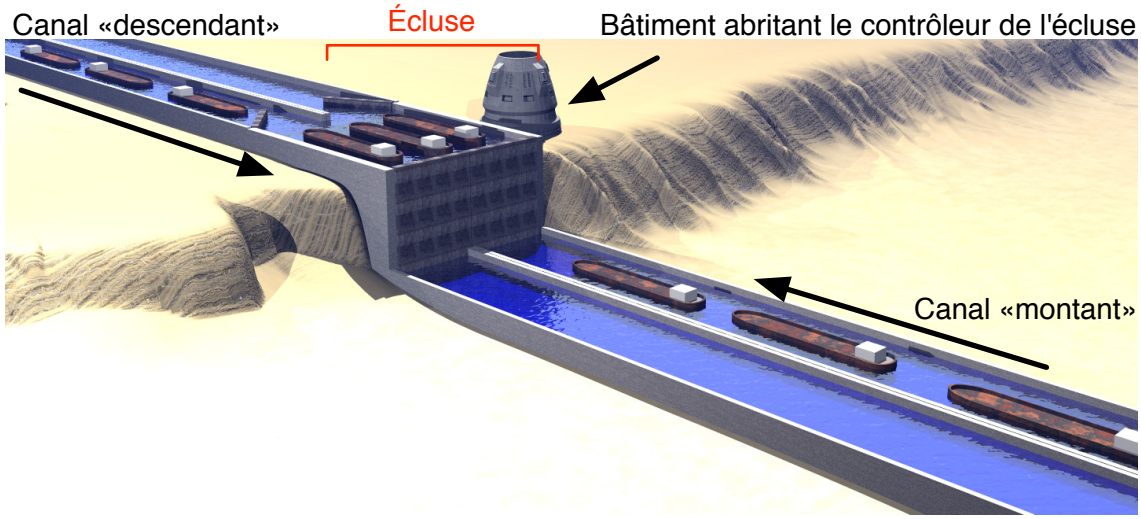

FIG. 1 – Architecture du système d'écluse

L'écluse monte et descend alternativement les péniches (car il n'y a qu'une seule écluse pour gérer les deux canaux). Pour assurer le passage de l'écluse dans les deux sens, le contrôleur change le sens de circulation après le passage d'un nombre de péniches égal à sa capacité. L'écluse ne change pas de position à vide ou avec un nombre de péniches inférieur à sa capacité.

L'écluse est implémentée au moyen d'une tâche qui synchronisera les péniches et leur permettra d'entrer dans l'écluse avant qu'elle ne monte (ou descende) puis d'en sortir lorsque le passage est terminé. Dans la suite de cet exercice, on considérera qu'il y a toujours assez de péniches pour remplir l'écluse.

On vous donne les structures de données suivantes :

```
-- Le sens de passage dans l'ecluse
type Sens is (Monter, Descendre);
-- La capacite de l'ecluse
Capacite : constant Natural := 3;
-- L'identite des peniches
subtype Identite is Integer range 1 .. Capacite * 3;

-- Le controleur de l'ecluse
task Controleur_Ecluse is
   -- point d'entree permetant a une peniche de demander de monter
```

```
   entry Monter (Qui : in Identite);
   -- point d'entree permetant a une peniche de demander de descendre
   entry Descendre (Qui : in Identite);
end Controleur_Ecluse;

-- Une peniche
task type Peniche is
   -- initialisation de la peniche (elle recoit son identite et sa direction)
   entry Init (Id : in Identite; S : in Sens);
   -- le controleur de l'ecluse signale a la peniche qu'elle peut repartir
   entry Sortir;
end Peniche;

-- Les peniches qui permettront de tester le systeme
Les_Peniches : array (Sens, Identite) of Peniche;
```

Pour passer l'écluse, les péniches doivent :

1. demander de monter (ou de descendre). Cette requête est bloquante tant que l'autorisation leur n'est pas donnée ;
2. entrer et ensuite attendre que l'autorisation de sortir leur soit donnée par le contrôleur de l'écluse ; lorsque cette autorisation est obtenue, on considère que la péniche sort effectivement de l'écluse.

Question 1 – 3 points

Écrivez le source du **body** d'une tâche `Peniche`.

Nous allons maintenant nous attacher à l'élaboration du contrôleur de l'écluse. Pour cela, nous déclarons dans le corps de la tâche `Controleur_Ecluse` les variables suivantes :

```
Sens_Courant : Sens    := Sens'First; -- sens courant de l'ecluse
Qui_Dedans   : array (1 .. Capacite) of Identite; -- identite des navires presents dans l'ecluse
```

Si, pour répondre aux questions suivantes, vous avez besoin de variables supplémentaires, vous devrez les déclarer et indiquer ce qu'elles représentent.

On donne également la procédure d'initialisation du système :

```
begin -- programme principal (Ecluse)
   for I in  Identite loop
      for J in  Sens loop
         Les_Peniches (J, I).Init (I, J);
      end loop;
   end loop;
end Ecluse;
```

Question 2 – 1 point

Indiquez la condition qui permet au contrôleur de laisser les péniches montantes entrer dans l'écluse.

Question 3 – 1 point

Indiquez la condition qui permet de déterminer lorsqu'il faut « changer de sens » (*i.e.* fermer les portes, amener les péniches de l'autre coté puis ouvrir l'écluse - ces actions ne seront pas identifiées dans le programme).

Question 4 – 0.5 point

Comment le contrôleur sait-il dans quelle direction (montante ou descendante) il doit trouver les péniches à qui signaler que le « voyage » est terminé ?

Question 5 – 6.5 points

Écrivez le source du **body** de la tâche `Controleur_Ecluse` pour que toutes les péniches puissent passer sans incident l'écluse et que toutes les tâches du programme se terminent correctement.

Exercice 2 – Questions de cours, 8 points

Considérons le programme suivant :

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Taches_1 is

   task type T1;
   task type T2;

   task body T1 is
   begin
      Put_Line ("Bonjour de T1");
   end T1;

   task body T2 is
      A : T1;
   begin
      Put_Line ("Bonjour de T2");
   end T2;

   Tab1 : array (1 .. 3) of T1;
   Tab2 : array (1 .. 2) of T2;

begin -- programme principal
   null;
end Taches_1;
```

Question 1 – 0.5 point

Combien de tâches ce programme comporte-t-il ?

Question 2 – 1 point

Quelles sont les conditions de terminaison de chacune des tâches du programme.

Question 3 – 1.5 point

Donnez un chronogramme de l'exécution de ce programme.

Considérons maintenant le programme ci dessous :

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Taches_2 is

   task T1 is
      entry Bonjour;
      entry Au_Revoir;
   end T1;
   task type T2;
   task type T3;

   task body T1 is
      Ap_Bjrs : Boolean := False;
   begin
      loop
         select when not Ap_Bjrs =>
            accept Bonjour;
            Ap_Bjrs := True;
            Put_Line ("Bonjour");
         or
            accept Au_Revoir;
            Ap_Bjrs := False;
            Put_Line ("Au revoir");
         end select;
      end loop;
   end T1;

   task body T2 is
   begin
      T1.Bonjour;
   end T2;

   task body T3 is
```

```
   begin
      T1.Au_Revoir;
   end T3;

   Tab2 : array (1 .. 5) of T2;
   Tab3 : array (1 .. 5) of T3;

begin -- programme principal
   null;
end Taches_2;
```

Question 4 – 1 point
Ce programme se termine-t-il ? justifiez brièvement votre réponse (5 lignes max).

Question 5 – 1 point
Proposez une modification qui change le comportement observé à la question précédente.

Question 6 – 2 points
Transformez `T1` en un objet protégé ayant un comportement équivalent.

Question 7 – 1 point
A) À quoi correspond l'espace mémoire alloué à une tâche créée dynamiquement via l'instruction `new` ?
B) Comment désalloue-t-on la mémoire associée à une tâche allouée dynamiquement ? (5 lignes max).