Université P. & M. Curie Examen Final
Licence Mention Informatique – L3 9 Janvier 2013
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 – Questions de cours (7 points)

Considérons le programme indiqué ci-dessous.

```
1  with Ada.Text_IO;
2  use  Ada.Text_IO;
3
4  procedure Chronogramme is
5
6     task type Tache (Id : Character) is
7        entry Entree;
8     end Tache;
9
10    type A_Tache is access Tache;
11
12    task body Tache is
13    begin
14       Put_Line ("Je suis le numero " & Id);
15       accept Entree;
16       Put_Line ("Adieu du numero " & Id);
17    end Tache;
18
19    Table_1 : array (1 .. 3) of A_Tache;
20    Table_2 : array (1 .. 2) of Tache ('1');
21
22 begin
23    Table_1 (3) := new Tache ('A');
24    declare
25       A : Tache ('B');
26    begin
27       Table_2 (2).Entree;
28       A.Entree;
29    end;
30    Table_1 (1) := new Tache ('2');
31    Table_1 (1).all.Entree;
32    declare
33       B : Tache ('4');
34       C : A_Tache := new Tache ('H');
35    begin
36       B.Entree;
37       Table_2 (1).Entree;
38       C.all.Entree;
39    end;
40    Table_1 (3).all.Entree;
41 end Chronogramme;
```

Question 1 – 1 point

Combien de tâches sont-elles créées dans ce programme ? justifiez brièvement votre réponse.

Question 2 – 3 points

Dessinez le chronogramme de l'exécution de ce programme en indiquant les numéros de lignes correspondant à la création des tâches et en imaginant que toute tâche qui peut avancer dans son exécution le fait aussitôt que possible.

On considère désormais le programme ci-dessous.

```
1  with Ada.Text_IO; use Ada.Text_IO;
2
3  procedure LE is
4
5     task Ecrivain;
6     task type Client;
7
8     protected Variable is
9        procedure Ecrire (Val : in Natural);
10       procedure Lire (Val : out Natural);
11    private
12       Valeur : Integer := -1;
13    end Variable;
14
15    task body Ecrivain is
16    begin
17       for X in 0 .. 10 loop
18          Variable.Ecrire (X);
19          delay 0.2;
20       end loop;
21    end Ecrivain;
22
23    task body Client is
24       Myval : Natural;
25    begin
26       for X in 0 .. 20 loop
27          Variable.Lire (Myval);
28          delay 0.15;
29       end loop;
30    end Client;
31
32    protected body Variable is
33
34       procedure Ecrire (Val : in Natural) is
35       begin
36          Valeur := Val;
37          Put_Line ("Erire" & Integer'Image (Valeur));
38       end Ecrire;
39       procedure Lire (Val : out Natural) is
40       begin
41          Put_Line ("   Lire" & Integer'Image (Valeur));
42          Val := Valeur;
43       end Lire;
44    end Variable;
45
46    C1, C2 : Client;
47
48 begin
49    null;
50 end LE;
```

Question 3 – 0,5 point

Donnez un enchaînement d'actions qui conduit à la levée de l'exception `Constraint_Error`.

Question 4 – 1 point

Quelle règle de synchronisation peut-on imposer pour éviter le problème ? Donnez le principe de la solution à mettre en œuvre.

Question 5 – 1,5 points

En supposant le problème corrigé, modifiez le code de ce programme pour que les écritures soient prioritaires par rapport aux lectures (*i.e.* une lecture ne commence que s'il n'y a pas d'écriture en cours ou en attente).

Exercice 2 – Autoroute automatique (13 points)

Nous considérons une section prototype d'autoroute automatique. L'architecture de ce système respecte le schéma de la figure 1. Les constituants sont :

- les véhicules automatisés,
- la chaussée divisée en sections.

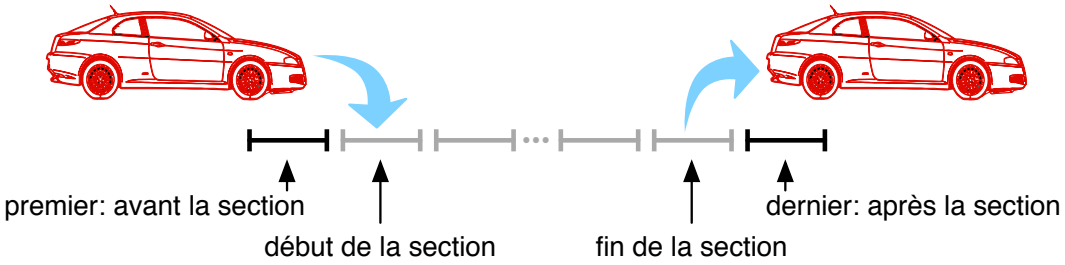

FIGURE 1 – Structure du système

Les sections de la chaussée sont numérotées de `Avant_Autoroute = Id_Segments'First` à `Apres_Autoroute = Id_Segments'Last` :

- Le premier segment correspond à une position « avant la section automatisée »,
- Le dernier segment correspond à une position « après la section automatisée ».

Les véhicules situés en dehors de la section automatisée utilisent ces positions pour indiquer qu'ils sont avant ou après la zone automatisée. Les autres valeurs correspondent à une position dans la section. On imaginera que les segments de l'autoroute sont numérotés de `Entree_Autoroute = Id_Segments'Succ (Id_Segments'First)` à `Sortie_Autoroute = Id_Segments'Pred (Id_Segments'Last)`.

Chaque véhicule est modélisé au moyen d'une tâche. On considère des « patrouilles » composées d'un véhicule de tête et de véhicules suiveurs. Les véhicules respectent les règles suivantes :

- les véhicules ne se dépassent pas ;
- le véhicule de tête peut décider d'avancer ou non (pour le décider, il fait appel à la fonction booléenne `Decider_d_Avancer`) lorsqu'il a la main ; lorsqu'il avance, il passe à la section suivante ;
- un véhicule suiveur ne peut avancer que si une section vide le sépare du véhicule qu'il suit. Pour cela, il doit demander à ce dernier sa position. Si la position de son prédécesseur le permet, il peut décider d'avancer ou non d'une section (toujours avec un appel à la fonction `Decider_d_Avancer`) ;
- tout véhicule qui atteint la dernière section de l'autoroute est automatiquement considéré comme sorti.

Pour construire ce système, on dispose des déclarations suivantes :

```
1   -- Déclaration de constantes (initialisation suivant les arguments de la ligne de commande)
2   Maxsegments  : constant Integer := Get_Initial_Value (1);
3   Max_Voitures : constant Integer := Get_Initial_Value (2);
4
5   -- Déclaration des types et constantes utiles pour le système
6   type Id_Segments is new Integer range 0 .. Maxsegments;
7   Avant_Autoroute  : constant Id_Segments := Id_Segments'First;
8   Entree_Autoroute : constant Id_Segments := Id_Segments'Succ (Id_Segments'First);
9   Sortie_Autoroute : constant Id_Segments := Id_Segments'Pred (Id_Segments'Last);
10  Apres_Autoroute  : constant Id_Segments := Id_Segments'Last;
11
12  type Id_Voitures is new Integer range 1 .. Max_Voitures;
13
14  type Une_Voiture;
15  type A_Une_Voiture is access Une_Voiture;
16
17  -- Fonction chargée de décider si un véhicule avance ou non.
18  function Decider_D_Avancer return Boolean;
```

On souhaite implémenter les véhicules sous forme de tâches. Ces tâches (`Une_Voiture`) sont créées dynamiquement et initialisées au moyen d'un point d'entrée `initialiser` qui prend en paramètre l'identificateur du véhicule et un pointeur vers le véhicule qui le précède (**`null`** dans le cas du véhicule de tête).

Lorsqu'un véhicule souhaite avancer, il consulte la position de celui qui le précède via un point d'entrée `Ma_Position` qui rend l'identificateur de la section du dans laquelle se trouve le véhicule interrogé.

Question 1 – 1 point
Écrivez le source de la spécification d'une tâche `Une_Voiture`.

L'ensemble des véhicules sont décrits dans une variable dont la déclaration est indiquée ci après :

```
1   Les_Voitures : array (Id_Voitures) of A_Une_Voiture;
```

Question 2 – 2 points
Écrivez le source du programme principal qui initialise la structure `Les_Voitures`. On considèrera que le véhicule i précède le véhicule $i + 1$.

Lorsque toutes les voitures automatisées sont sorties de l'autoroute, il faut terminer les tâches du système. Pour cela, on utilise une variable globale `Compteur_Sorties`, permettant de compter les véhicules ayant quitté l'autoroute automatique.

Question 3 – 1 point
Que comportement incorrect pourrait-on mettre en évidence si on choisissait de terminer une tâche `Une_Voiture` dès que le véhicule qu'elle représente est sorti de l'autoroute ?

Question 4 – 1 point
Doit-on protéger l'accès à la variable `Compteur_Sorties` ? Justifiez votre réponse

Chaque véhicule quittant l'autoroute se signale au moyen du service `Je_Suis_Sorti` et la terminaison des tâches `Une_Voiture` est testée au moyen du service `Tout_Le_Monde_Est_Sorti`.

Question 5 – 1 point
Écrivez la spécification de la structure de données gérant `Compteur_Sorties`.

Question 6 – 2 points
Écrivez le source implémentant les services associés à `Compteur_Sorties`.

Chaque véhicule, une fois initialisé, tente de rentrer sur l'autoroute et le fait dès que cela est possible tout en respectant les conditions de progression définies plus haut. Ensuite, il progresse jusqu'à quitter l'autoroute. Enfin, la tâche modélisant le véhicule attend que l'ensemble des véhicules de la patrouille soient sortis pour se terminer.

Question 7 – 5 points
Écrivez le source implémentant une tâche de type `Une_Voiture`.