Université P. & M. Curie Examen
Licence Mention Informatique – L3 20 Janvier 2011
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 – Systèmes de navettes (14 points)

Considérons des navettes automatisées circulant sur une voie ferrée circulaire dont nous donnons l'architecture en figure 1. Le système est divisé en 16 segments, chacun ayant une taille suffisante pour stocker une navette.

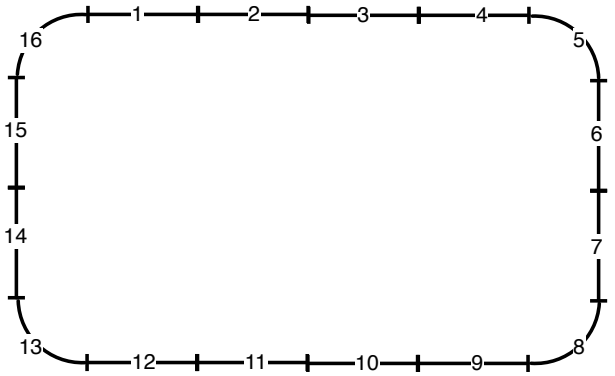

FIGURE 1 – Structure des voies de circulation des navettes

Le système est décrit au moyen des éléments suivants :

```
-------------------------------------------------------------------------
-- Les constantes (plus une exception en cas de problème)
-------------------------------------------------------------------------
Catastrophe : exception;
Max_Segment : constant Natural := 16;
Nb_Navettes : constant Natural := 7;

-------------------------------------------------------------------------
-- Les types de base
-------------------------------------------------------------------------
subtype Id_Segment is Natural range 1 .. Max_Segment;
subtype Id_Navette is Natural range 1 .. Nb_Navettes;
type Segment_Status is (Libre, Occupe, Tampon);

-------------------------------------------------------------------------
-- Autres types
-------------------------------------------------------------------------
type Navette;
type Segment;
type A_Navette is access Navette;
type A_Segment is access Segment;
-------------------------------------------------------------------------
-- Les variables globales
-------------------------------------------------------------------------
Tab_Navettes    : array (Id_Navette) of A_Navette; -- Les navettes
Dernier_Segment : A_Segment; -- Référence vers le "dernier" segment (le numéro 16)
```

IMPORTANT : *Les navettes circulent dans le sens des aiguilles d'une montre. Pour assurer la fiabilité du système, il doit toujours y avoir au moins un segment inoccupé entre deux navettes qui se suivent. Un segment occupé par une navette est donc toujours suivi d'un segment tampon.*

Les segments sont décrits au moyen du type protégé `Segment(Id : Id_Segment)` encapsulant les données suivantes :

- `Etat_Segment` qui caractérise l'état du segment : `Libre` (une navette peut donc entrer), `Tampon` (le segment n'est pas occupé par une navette mais constitue la zone "tampon" de sécurité) ou `Occupe` (le segment est occupé par une navette),
- `Succ` qui référence le successeur du segment dans le chemin de fer circulaire.

Le type protégé `Segment` offre également plusieurs services :

- `Init (Succes : in A_Segment)` qui permet d'affecter son successeur au segment,
- `Successeur` qui renvoie un pointeur sur le segment successeur dans le chemin de fer circulaire,
- `Entrer` qui autorise une navette à entrer dans le segment,
- `Quitter` qui permet à une navette de signaler qu'elle a quitté le segment,
- `S_Eloigner` qui permet à une navette de signaler qu'elle a quitté le successeur du segment.

Question 1 – 2 points

Écrivez la spécification du type protégé `Segment`. Vous justifierez, pour chacun de ses services, la catégorie que vous lui aurez choisie.

Question 2 – 2 points

Écrivez le source du corps du type protégé `Segment`.

Question 3 – 3 points

Écrivez le source de la partie du programme principal qui construit le chemin de fer circulaire (*i.e.* les liaisons entre les segments) et initialise les segments.

Nous représentons les navettes au moyen du type tâche `Navette` dont la spécification est donnée ci dessous.

```
task type Navette (Id : Id_Navette) is
   -- Demander a une navette d'entrer dans le segment 1 de la voie ferrée
   entry Entrer_Dans_Reseau;
   -- Demander à une navette de quitter le réseau
   entry Quitter_Reseau;
end Navette;
```

Une tâche `Navette` se comporte comme suit :

- attendre un ordre pour s'engager sur le segment 1,
- répéter ensuite :
  - Entrer dans le segment suivant
  - Mettre à jour l'état des segments précédents
- jusqu'à quitter le réseau : la sortie est exécutée si la navette en a reçu l'ordre via le point d'entrée `Quitter_Reseau` et si sa position le permet (elle est sur le segment de numéro maximum).

Question 4 – 2 points

Une navette qui entre sur le réseau exécute la séquence suivante :

```
Dernier_Segment.all.Entrer;
Seg_Courant := Dernier_Segment.all.Successeur;
Seg_Courant.all.Entrer;
Seg_Tampon := Dernier_Segment;
Seg_Tampon.all.Quitter;
```

Expliquez pourquoi il est nécessaire d'exécuter `Dernier_Segment.all.Entrer`. Que peut-il se passer si cet appel est déplacé après `Seg_Courant.all.Entrer` ?

Question 5 – 4 points

Écrivez le source du corps de la tâche `Navette`. Pour cela, vous pouvez utiliser les variables suivantes, locales à chaque navette :

```
Seg_Tampon, Seg_Courant, Seg_Suivant : A_Segment := null;
Sortir_du_Reseau : Boolean                       := False;
```

Question 6 – 1 point

Le système crée et lance sur le réseau ferré 8 navettes qu'il laisse tourner. Quel problème peut se poser ? Expliquez.

Exercice 2 – Remaniement du système de navettes (6 points)

On reprend les éléments du système précédent mais en supprimant la synchronisation via le type protégé `Segment`. Désormais, ce sont les navettes qui contiennent dans leur contexte le numéro du segment dans lequel elles se trouvent. Il est possible de communiquer avec une navette au moyen de deux mécanismes de rendez-vous permettant :

1. de transmettre à une navette un pointeur sur la navette qui la précède (initialisation),
2. de demander à une navette le numéro du segment sur lequel elle se trouve.

Lors de sa création, chaque navette est placée sur le segment $2 \times Id$ où Id représente son identité (de type `Id_navette`) qui lui est transmise par un discriminant. Nous supposons que le système comporte au moins deux navettes.

Question 1 – 1 point
Donnez la spécification d'une tâche navette.

Une navette gère sa propre position. Pour avancer, elle doit demander à la navette qui se situe devant elle sa position et s'assurer qu'il restera bien un segment d'écart entre les deux avant d'avancer. On se propose de programmer une tâche `Navette` de la manière suivante :

- Attendre l'initialisation
- Boucler sur
  - Demander sa position à mon prédécesseur et avancer si c'est possible
  - Attendre qu'on me demande ma position

Question 2 – 2 points
Nous considérons un système avec 3 navettes dont les identifiants sont 1, 2 et 3. Représentez par un chronogramme le comportement des trois navettes une fois qu'elles ont été créées et initialisées. Expliquez ce que vous constatez.

Question 3 – 1 point
Dans la programmation qui vous est proposée, une navette peut-être bloquée en attendant qu'on lui demande sa position. Proposez une solution permettant de rendre le système non bloquant lors de ce rendez-vous, i.e. la navette attendra bien qu'on lui demande sa position mais ne sera pas bloquée si cette demande n'arrive pas (ou met trop de temps à arriver).

On s'intéresse au corps d'une tâche `Navette` et, pour le programmer, on supposera l'existence des fonctions ci après :

```
-- Rend TRUE si la distance entre Pos et Pos_Devant permet a la navette en position Pos d'avancer
function Distance_Ok (Pos, Pos_Devant : in Id_Segment) return Boolean;
-- Rend le segment suivant le segment Pos
function Segment_suivant (Pos : in Id_Segment) return Id_Segment;
```

Question 4 – 2 points
Proposez une programmation de la tâche `Navette` non bloquante (i.e. l'exécution du programme ne bloque pas, ce qui ne veut pas dire que les navettes avancent obligatoirement !). **Attention**, une navette ne disposant pas de suffisamment d'information pour garantir la fiabilité du système n'avancera pas.