Université P. & M. Curie Examen Final
Licence Mention Informatique – L3 15 Décembre 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 – Pilotage de la capote d'un cabriolet (5 points)

La firme automobile Jussieu met au point son nouveau roadster LI330 (voir figure 1) pour lequel il faut concevoir le système de tâche permettant de gérer l'ouverture et la fermeture de la capote. Nous nous intéressons plus précisément à son ouverture (on parlera de replier la capote dans le coffre).

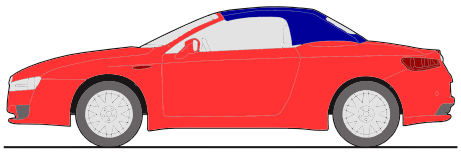

FIGURE 1 – Le nouveau roadster LI330 de chez Jussieu

La procédure d'ouverture de la capote respecte les contraintes suivantes :
- si les vitres sont levées, les descendre,
- une fois les vitres descendues, ouvrir le coffre par l'avant afin de dégager l'espace pour rentrer la capote,
- une fois le coffre ouvert par l'avant, activer le mécanisme de repliage de la capote dans le coffre,
- une fois la capote repliée, fermer le coffre sur la capote repliée,
- une fois le coffre refermé, si les vitres étaient levées au début de la procédure, les remonter.

Le système est décrit par les structures indiquées ci-dessous :

```
1   -- determiner le cote du véhicule
2   type Le_Cote is (Gauche, Droite);
3
4   -- monter et descendre les vitres
5   task type Moteur_Vitre_Monter (C : Le_Cote) is
6      entry termine; -- s'active quand l'operation est terminee
7   end Moteur_Vitre_Monter;
8   task type Moteur_Vitre_Descendre (C : Le_Cote) is
9      entry termine; -- s'active quand l'operation est terminee
10  end Moteur_Vitre_Descendre;
11
12  type A_Moteur_Vitre_Monter is access Moteur_Vitre_Monter;
13  type A_Moteur_Vitre_Descendre is access Moteur_Vitre_Descendre;
14
15  -- ouvrir et fermer le coffre par l'avant
16  task type Coffre_Par_L_Avant_Ouvrir;
17  task type Coffre_Par_L_Avant_Fermer;
18
19  -- replier la capote
20  task type Capote_Replier;
21
22  -- variable permettant de connaître l'état des fenêtres
23  Vitres_Levees : array (Le_Cote) of Boolean;
24
25  -- pointeurs pour démarrer les actions des vitres
26  Monter_Une_Vitre_G, Monter_Une_Vitre_D : A_Moteur_Vitre_Monter;
27  Descendre_Une_Vitre_G, Descendre_Une_Vitre_D : A_Moteur_Vitre_Descendre;
```

Les tâches représentent des moteurs. On considère que la création d'une tâche déclenche le moteur correspondant jusqu'à ce qu'il ait terminé son action. Les variables sont utilisables pour connaître l'état du système ou faciliter la création de certaines tâches.

Question 1 – 2 points
Dessinez le chronogramme de l'exécution du système en considérant que les deux vitres sont initialement levées. Le chronogramme doit faire figurer, outre les synchronisations, toutes les tâches impliquées dans l'opération, l'instant où elles démarrent et l'instant où elles se terminent.

Question 2 – 3 points
Ecrivez le programme principal qui respecte le mode opératoire défini, en permettant le déroulement parallèle des tâches dès que c'est possible. Vous ne vous préoccuperez pas de l'initialisation de la variable `Vitres_Levees` et vous considèrerez que le corps des tâches, déjà programmé, réagit comme indiqué sur les spécifications correspondantes.

Exercice 2 – Gestion d'un hôtel (15 points)

Nous nous intéressons à la gestion d'un hôtel et, plus particulièrement, à celle des cartes-clefs générées automatiquement à l'arrivée des clients. Le système que nous étudions est composé :

- de clients,
- de guichetiers
- des serrures (une par chambre),
- du système de gestion des clefs de l'hôtel.

Un **client** entre dans l'hôtel, contacte un guichetier au hasard (via l'entrée `Check_In`) qui lui attribue une chambre et la carte qui permet de l'ouvrir. Il se rend ensuite dans cette chambre, en ressort et y rentre un certain nombre de fois (aléatoire) avant de quitter l'hôtel en rendant sa carte (via l'entrée `Check_Out`) à un guichetier, lui aussi choisi aléatoirement.
Un **guichetier** interagit avec les clients et effectue ses tâches (attribuer une clef, récupérer une clef) en utilisant le système de gestion des clefs de l'hôtel.
Une **serrure** permet de déclencher l'ouverture de la porte si la carte entrée par le client est "correcte". Une serrure mémorise une valeur que l'on nommera `Courante` et une carte mémorise deux valeurs que l'on nommera `Ancienne` et `Nouvelle`. Le principe de fonctionnement est le suivant lorsqu'une carte est insérée dans une serrure :

- si `Ancienne` = `Courante` le client entre dans sa chambre pour la première fois, `Courante` prend la valeur de `Nouvelle`.
- sinon, l'entrée n'est autorisée que si les valeurs de `Courante` et `Nouvelle` sont identiques.

Le **système de gestion des clefs de l'hôtel** offre au guichetier les opérations suivantes :

- `Initialise` qui est invoquée par les guichetiers pour initialiser le système de gestion des clef *mais qui n'a d'effet que pour le premier appel*,
- `Genere_Carte` qui permet à un guichetier d'attribuer une chambre à un client et d'obtenir la carte qui autorisera son ouverture,
- `Libere_Chambre` qui permet à un guichetier de déclarer qu'une chambre est libre.

L'hôtel est décrit par le programme ci dessous.

```
1   -- Pour la generation de tout ce qui est aleatoire
2   package Alea_Int is new Ada.Numerics.Discrete_Random (Integer);
3   Seed : Alea_Int.Generator;
4   function Calcule_Attente return Duration is
5   begin
6      return Duration (Alea_Int.Random (Seed) mod 10) * 0.1;
7   end Calcule_Attente;
8
9   -- Constantes pour dimentionner le système
10  Nb_Chambres   : constant Positive := 4;
11  Nb_Clients    : constant Positive := 8;
12  Nb_Guichetiers : constant Positive := 2;
```

```
13    Clef_Par_Defaut : constant Natural := 0;
14
15    -- Les types identifiant chambres, clients et guichetier
16    subtype Id_Chambre is Natural range 1 .. Nb_Chambres;
17    subtype Id_Client is Natural range 1 .. Nb_Clients;
18    subtype Id_Guichetier is Natural range 1 .. Nb_Guichetiers;
19     -- Une clef electronique sur la carte
20    subtype Clef is Natural;
21    -- Les donnees pour gérer une chambre
22    type Donnees_Chambre is
23       record
24          Est_Libre     : Boolean := True;
25          Derniere_Clef : Clef;
26       end record;
27    type Tab_Chambres is array (Id_Chambre) of Donnees_Chambre;
28
29    -- Une carte faisant office de clef
30    type Une_Carte is
31       record
32          Ancienne : Clef;
33          Nouvelle : Clef;
34       end record;
35    -- Les interlocuteurs dynamiques du système
36    type Client;
37    type A_Client is access Client;
38    type Guichetier;
39    type A_Guichetier is access Guichetier;
40    type Serrure;
41    type A_Serrure is access Serrure;
42
43    -- La gestion du système de clefs de l'hôtel
44    protected Systeme_De_Gestion_Des_Clefs is
45       -- L'entrée suivante initialise le système uniquement lors de la première invocation
46       procedure Initialise;
47       entry Genere_Carte (Pour  : out Id_Chambre;
48                           Carte : out Une_Carte);
49       procedure Libere_Chambre (Num : in Id_Chambre);
50    private
51       Les_Chambres: Tab_Chambres;
52       Init_Faite: Boolean := False;
53       Chambres_Libres : Natural := Nb_Chambres;
54    end Systeme_De_Gestion_Des_Clefs;
55
56    -- les tâches correspondant aux acteurs du système
57    task type Guichetier (Id : Id_Guichetier) is
58       entry Check_In (Moi     : in A_Client;
59                       Chambre : out Id_Chambre;
60                       Carte   : out Une_Carte);
61       entry Check_Out (Moi     : in A_Client;
62                        Chambre : in Id_Chambre);
63    end Guichetier;
64
65    task type Client (Id : Id_Client) is
66       entry Init (Me : in A_Client); -- Transmettre son pointeur à un client
67       entry Entre; -- moyen par lequel une serrure autorise un client à rentrer
68    end Client;
69
70    task type Serrure (Num : Id_Chambre) is
71       entry Demande_Ouverture (Moi  : in A_Client;
72                                Avec : in Une_Carte);
73    end Serrure;
74
75    -- les variables globales du système
76    Les_Guichetiers : array (Id_Guichetier) of A_Guichetier;
77    Les_Chambres    : array (Id_Chambre) of A_Serrure;
78    P               : A_Client;
```

Question 1 – 1 point

Pourquoi utilise-t-on un objet protégé plutôt qu'un paquetage pour programmer le système de gestion des clés ?

Question 2 – 0,5 point

Combien de tâches le système comporte-t-il au **`begin`** du programme principal sachant qu'aucune autre entité n'est déclarée entre la ligne 78 du listing présenté et le **`begin`** en question ?

Question 3 – 1,5 point
Écrivez le source du programme principal qui lance les différents interlocuteurs du système (`Nb_Clients` clients, `Nb_Guichetiers` guichetiers et `Nb_Chambres` serrures de chambres).

Question 4 – 2 points
Écrivez le source d'une tâche de type `Client` en considérant qu'il rentre dans sa chambre deux fois.

Question 5 – 3 points
Écrivez le source d'une tâche de type `Serrure` en suivant le protocole désigné. Vous considèrerez que la serrure réveille le client sur le point d'entrée `Entre` (il a transmis son pointeur à la serrure) uniquement lorsque la carte rentrée est la bonne.

Question 6 – 2 points
Écrivez le source d'une tâche de type `Guichetier` en utilisant le `Systeme_De_Gestion_Des_Clefs`.

Question 7 – 3 point
Écrivez le source du `Systeme_De_Gestion_Des_Clefs`.

Question 8 – 1 point
Nous considérons un système avec un guichetier, deux chambres et deux clients C_1 et C_2. C_1 obtient la chambre 1 et C_2 la chambre 2 mais se trompe de porte et souhaite entrer dans la chambre 1. Que se passe-t-il ? Dessinez le chronogramme du système (vous vous arrêterez à la fin du scénario décrit).

Question 9 – 1 point
Nous considérons un système avec un guichetier, deux chambres et deux clients C_1 et C_2. C_1 obtient la chambre 1 mais quitte l'hôtel et rend sa clé sans jamais être entré dans la chambre. Le Client C_2 obtient ensuite la même chambre et insère sa carte dans la serrure. Que se passe-t-il ? Dessinez le chronogramme du système (vous vous arrêterez à la fin du scénario décrit).