Université P. & M. Curie Examen
Licence Mention Informatique – L3 3 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 – Le restaurant (17,5 points)

Considérons un restaurant dans lequel nous avons les éléments suivants (voir l'architecture du système présentée en figure 1) :

- Des clients qui s'installent par groupe de `Nb_Clients_Par_Table` autour de tables, attendent que le maître d'hôtel vienne prendre la commande, puis qu'un serveur amène celle-ci. Ensuite, ils mangent puis payent et quittent le restaurant,
- Un maître d'hôtel qui reçoit les clients et les installe à leur table. Lorsque celle-ci est pleine, il prend leurs commandes. À la fin du repas, il reçoit le paiement.
- Des cuisiniers qui préparent les menus en consultant les commandes et les déposent sur une file de menus,
- Des serveurs qui distribuent les menus pris depuis la file de menus aux clients concernés.

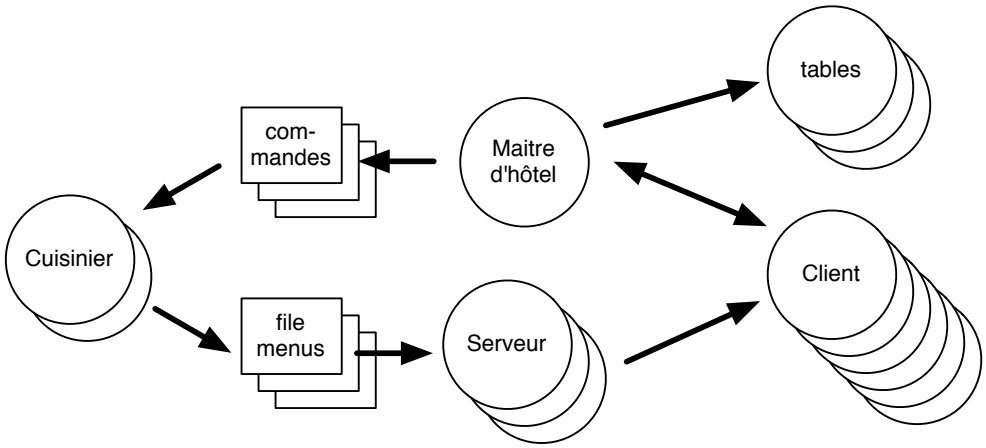

FIGURE 1 – Architecture du système

La figure 1 indique également les schémas de communication entre les différentes entités du système. Le sens des flèches indique l'initiateur de la communication mais aussi l'interlocuteur destinataire de la dite communication.
Le listing ci-après donne dans un premier temps l'ensemble des déclarations "utilitaires" pour le système.

```
-----------------------------------------------------------------------
-- Les constantes
-----------------------------------------------------------------------
Nb_Cuisiniers        : constant Positive := 2;
Nb_Serveurs          : constant Positive := 3;
Nb_Tables            : constant Positive := 3;
Nb_Clients_Par_Table : constant Positive := 3;
Nb_Clients           : constant Positive := Nb_Tables * Nb_Clients_Par_Table;
Nb_Menus             : constant Positive := 3;
Max_Commandes        : constant Positive := 3;
Max_File_Menus       : constant Positive := 3;
```

```
-------------------------------------------------------------------------
-- Les types de base
-------------------------------------------------------------------------
type Id_Cuisiniers is range 1 .. Nb_Cuisiniers;
type Id_Serveurs is range 1 .. Nb_Serveurs;
type Id_Tables is range 1 .. Nb_Tables;
type Id_Clients is range 1 .. Nb_Clients;
type Id_Menus is range 1 .. Nb_Menus;

-------------------------------------------------------------------------
-- Autres types
-------------------------------------------------------------------------
-- Les tâches dynamiques du système (définition incomplète)
type Client;
type A_Client is access Client;
type Serveur;
type A_Serveur is access Serveur;
type Cuisinier;
type A_Cuisinier is access Cuisinier;
-- types "de service"
type Tab_A_Client is array (1 .. Nb_Clients_Par_Table) of A_Client;
type Tab_Menus is array (1 .. Nb_Clients_Par_Table) of Id_Menus;
type Une_Commande is record
   Les_Clients   : Tab_A_Client;
   Les_Commandes : Tab_Menus;
end record;
type Table_Commandes is array (1 .. Max_Commandes) of Une_Commande;
type Tab_Clients is array (1 .. Max_File_Menus) of A_Client;
-- Une table
type Table is record
   Clients_Arrives : Natural      := 0;
   Les_Clients     : Tab_A_Client := (others => null);
end record;
```

Nous déclarons ensuite les tâches du système

```
task type Client (Id : Id_Clients; My_Table : Id_Tables) is
   -- Initialisation. Me transmet la valeur du pointeur sur la tâche concernée.
   entry Init (Me : in A_Client);
   -- Commander (au maître d'hôtel)
   entry Commander (My_Menu : out Id_Menus);
   -- recevoir son plat des mains d'un serveur
   entry Recevoir_Plat;
end Client;

task Maitre_D_Hotel is
   -- Accueillir un client et lui affecter sa table réservée
   entry Accueillir (Who : in A_Client; For_Table : in Id_Tables);
   -- Recevoir le payement d'un client
   entry Payer (Me : in Id_Clients; Table : in Id_Tables);
end Maitre_D_Hotel;

task type Cuisinier (Id : Id_Cuisiniers);

task type Serveur (Id : Id_Serveurs);
```

Question 1 – 0,5 point

Expliquez pourquoi `Maitre_D_Hotel` est une tâche et non un type tâche ? Expliquez brièvement quelles sont les conséquences sur les conditions du démarrage de `Maitre_D_Hotel`.

Nous disposons des variables globales suivantes :

```
Les_Tables      : array (Id_Tables) of Table;
Les_Clients     : array (Id_Clients) of A_Client;
Les_Serveurs    : array (Id_Serveurs) of A_Serveur;
Les_Cuisiniers  : array (Id_Cuisiniers) of A_Cuisinier;
```

Le programme principal doit lancer les tâches sachant qu'un client c se voit affecter la table numéro $c \ mod$ `Nb_Tables + 1`.

Question 2 – 0,5 point
Combien de tâches le système comporte-t-il juste avant l'exécution de la première instruction du programme principal ?

Question 3 – 1,5 point
Écrivez le source du programme principal qui crée et initialise les tâches du système.

Pour écrire la suite, nous disposons de la fonction suivante pour choisir aléatoirement un menu :

```
function Choisir_Menu return Id_Menus; -- fonction de choix aléatoire d'un menu
```

Question 4 – 2,5 points
Écrivez le source d'une tâche `Client` (voir le comportement défini au début du problème).

Nous introduisons maintenant les éléments suivants dans le système :

```
protected type File_Menu is
   entry Deposer_Menu (Pour : in A_Client);
   entry Retirer_Menu (Pour : out A_Client);
private
   Les_Menus        : Tab_Clients;
   Taille_File_Menu : Natural := 0;
end File_Menu;

protected Commandes is
   -- debut de transaction sur une saisie de commande
   entry Debuter_Commande;
   -- saisie de la commande d'un convive
   entry Rajouter_Commande (Menu : in Id_Menus; Pour : in A_Client);
   -- fin de transaction sur une saisie de commande
   procedure Finir_Commande;
   -- retirer la commande d'une table
   entry Retirer_Une_Commande (Cmd : out Une_Commande);
private
   Nb_Commandes    : Natural := 0;
   Les_Commandes   : Table_Commandes;
   Saisie_En_Cours : Boolean := False;
   Crt_Client      : Positive;
end Commandes;
```

Question 5 – 0,5 points
Qu'assure la notion de `procedure` dans un objet/type protégé ? Indiquez également la différence entre une `entry` et une `procedure`.

La gestion des commandes se fait suivant un mécanisme transactionnel différent selon que l'on écrive la commande (le maître d'hôtel) ou qu'on l'exécute (les cuisiniers) :

- **Ecriture :** le maître d'hôtel démarre la transaction, prend la commande de chacun des convives de la table qu'il traite, puis termine la transaction.
- **lecture :** un cuisinier accède à une commande complète et confectionne les menus commandés les uns après les autres.

Notons que le système contient au plus `Max_Commandes` et que les deux types de transaction sont en exclusion mutuelle (i.e. aucune commande ne peut-être retirée pendant qu'on en écrit une).

Question 6 – 2 points
Écrivez le corps de l'objet protégé `Commandes`.

Rappelons que le maître d'hôtel affecte les clients aux tables qu'ils ont réservées et ne prend la commande, suivant le protocole défini plus haut, que lorsque tous les clients sont arrivés. Lorsqu'elle n'a plus d'utilité, la tâche `Maitre_D_Hotel` se termine.

Question 7 – 3 points
Écrivez le corps de la tâche `Maitre_D_Hotel`.

Nous allons maintenant programmer les tâches `Cuisinier` et `Serveur` sans nous préoccuper de leur terminaison. Notons que les cuisiniers transmettent le pointeur du client dont ils ont confectionné le plat via la file de menus dans des files de menus déclarées comme suit :

```
Les_Files_De_Menus : array (Id_Menus) of File_Menu;
```

Question 8 – 2 points
Écrivez le corps d'une tâche `Cuisinier`.

Question 9 – 2 points
Écrivez le corps d'une tâche `Serveur`.

Nous nous focalisons sur l'exécution du système en considérant deux clients, un cuisinier, un serveur et une table de deux places. Vous ne gèrerez qu'un seul type de menu.

Question 10 – 3 points
Construisez le chronogramme du système jusqu'à ce que les clients quittent le restaurant. Vous indiquerez explicitement les créations dynamiques de tâche.

Exercice 2 – Questions de cours (2,5 points)

Nous considérons la tâche suivante :

```
task Sem_Priv is
   entry P;
   entry V;
end Sem_Priv;

task body Sem_Priv is
   Occcupe : Boolean := False;
begin
   loop
      select when not Occcupe =>
         accept P;
         Occcupe := True;
      or
         accept V;
         Occcupe := False;
      or
         terminate;
      end select;
   end loop;
end Sem_Priv;
```

Question 1 – 1,5 point
Transformez cette tâche en type protégé dont vous donnerez la déclaration et le corps. Vous justifierez les choix de traduction des entrées de la tâche.

Question 2 – 1 point
Comparez la désallocation d'une tâche et d'une variable d'un autre type allouées dynamiquement toutes les deux.