Université P. & M. Curie Examen Partiel
Licence Mention Informatique – L3 24 Octobre 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 – Chaîne de construction (16 points)

On considère une chaîne qui se compose de trois classes d'entités. Un *producteur* qui produit des colis, un *Aiguillage* qui les dispatche sur $N =$`Max_Sorties` *Sorties* (voir Figure 1). Les colis disposent tous d'un identificateur compris entre 1 et `Max_Colis`. La règle qu'applique le composant *Aiguillage* est de transmettre le colis d'identité C à la sortie $S = C \mod$ `Max_Sorties`$+1$.

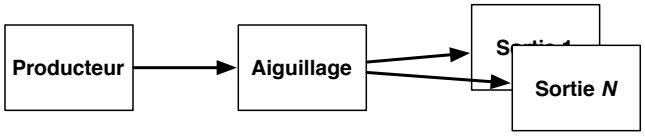

FIGURE 1 – Architecture du système

Le système est décrit au moyen des éléments suivants :

```
1   ------------------------------------------------------------------------
2   -- Les constantes
3   ------------------------------------------------------------------------
4   Max_Colis   : constant Natural := 10;
5   Max_Sorties : constant Natural := 3;
6
7   ------------------------------------------------------------------------
8   -- Les types de base
9   ------------------------------------------------------------------------
10  subtype Id_Colis is Integer range 1 .. Max_Colis;
11  subtype Id_Sorties is Integer range 1 .. Max_Sorties;
12
13  ------------------------------------------------------------------------
14  -- Tâches
15  ------------------------------------------------------------------------
16  task Producteur;
17
18  task type Sortie  is
19     entry Initialise (Id : in Id_Sorties);
20     entry Prends_Colis (Idc : in Id_Colis);
21  end Sortie;
22
23  task Aiguillage is
24     entry Transmets_Colis (Idc : in Id_Colis);
25  end Aiguillage;
26
27  ------------------------------------------------------------------------
28  -- Les variables globales
29  ------------------------------------------------------------------------
30
31  Les_Sorties : array (Id_Sorties) of Sortie;
```

Question 1 – 1 point
Si l'on considère les constantes données dans le listing décrivant le système, combien de tâches comporte le programme ?

Question 2 – 1,5 point
Quelles seraient les modifications à apporter au programme si on choisissait d'identifier les différentes tâches `Sorties` au moyen d'un discriminant ?

Question 3 – 1 point
Écrivez le source du programme principal qui initialise le système de tâches de la chaîne.

Question 4 – 1 point
Écrivez le source du `Producteur` qui crée `Max_Colis` et les transmet à l'`Aiguillage` avant de se terminer.

Question 5 – 2,5 points
Écrivez le source d'une `Sortie` qui, après initialisation, récupère les colis en provenance de l'`Aiguillage` tant que celui-ci en a à transmettre, puis se termine.

Question 6 – 3 points
Écrivez le source d'`Aiguillage` qui, tant que le `Producteur` envoie des colis, les dispatche à la bonne tâche `Sortie`, puis se termine.

On modifie maintenant l'architecture du système en recyclant les colis. Cela correspond à faire "boucler" le système : les tâches de type `Sortie` transmettant, une fois le colis déballé, l'emballage vide à la tâche `Producteur` qui les remplit et les renvoie dans le système.

Question 7 – 1 point
Quelle(s) est(sont) la(les) tâche(s) dont la spécification doit être modifiée pour permettre ce nouveau fonctionnement ? Justifiez et donnez la(les) nouvelle(s) spécification(s).

Question 8 – 1 point
Écrivez la nouvelle version du source de `Sortie` afin de gérer le recyclage.

Question 9 – 3 points
Écrivez la nouvelle version du source de `Producteur` afin de gérer le recyclage tout en maximisant l'activité du `Producteur` : celui-ci ne doit se bloquer que s'il n'y a pas d'emballage disponible dans le système.

Question 10 – 1 point
Peut-il y a voir un interblocage dans le système dans le système que vous venez de programmer ? Justifiez.

Exercice 2 – Questions de cours (4 points)

Considérons le programme ci dessous.

```
1  with Ada.Text_Io;
2  use Ada.Text_Io;
3
4  procedure Chronogramme is
5
6     task type C;
7
8     task type S is
9        entry S;
10    end S;
11
12    A  : C;
13    Ts : array (1 .. 2) of S;
```

```
14
15    task body C is
16
17    begin
18       for I in Ts'range loop
19          Ts(I).S;
20          Put_Line("Un S de traite");
21       end loop;
22    end C;
23
24    task body S is
25
26    begin
27       loop
28          select
29             accept S;
30          or
31             terminate;
32          end select;
33       end loop;
34    end S;
35
36 begin -- Chronogramme
37    Put_Line("phase 1");
38    declare
39       C_1 : C;
40    begin
41       null;
42    end;
43    Put_Line("phase2");
44    declare
45       C_2 : C;
46    begin
47       null;
48    end;
49 end Chronogramme;
```

Question 1 – 2,5 points

Construisez le chronogramme associé à ce programme en considérant que les commutations se font lorsque les tâches sont bloquées uniquement. Vous devrez indiquer les actions qui peuvent être effectuées en parallèle. Vous expliciterez également ce qui est strictement séquentiel.

Considérons maintenant la nouvelle séquence de code ci-dessous, située dans le corps d'une tâche qui effectue l'appel d'un point d'entrée d'une tâche `Le_Serveur` dans les conditions suivantes.

```
1 loop
2    select
3        Le_Serveur.Appel;
4    or
5       terminate;
6    end select;
7 end loop;
```

Question 2 – 1,5 points

Que pensez-vous d'un tel usage du **`select`** ? commentez et justifiez votre position.