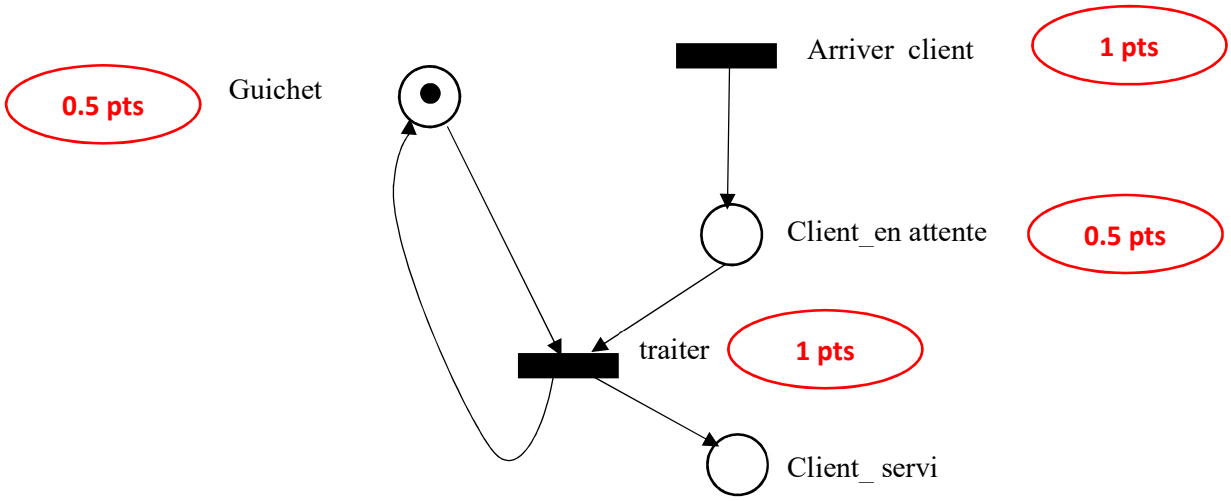
University of Larbi Ben Mhidi Oum El Bouaghi
Faculty of Exact Sciences and Life and Natural Sciences
Department of Mathematics and Computer Science

Study stream : Computer Science
Level: 1st Year Master AD
Year: 2025/2026
Duration: 1H:30mn

Corrigé type examen outils formels. **Date : 2026.**

Exercice 01 :



Exercice 02 :

```
\begin{zed}
  voiture==\nat

\end{zed}
```

0.5 pts

```
\begin{schema}{parking}
 courant: \power voiture\\
 capacite: \nat
\where
#courant\leq capacite

\end{schema}
```

1 pts

0.5 pts

```
\begin{schema}{entrer}
\Delta parking \\
 v?: voiture
\where
#courant<capacite \land
v? \notin courant\\
courant'= courant \cup \{v?\}

\end{schema}
```

1 pts

```
\begin{schema}{sortir}
\Delta parking\\
 v?: voiture
```

1 pts

```
\where

v? \in courant\\
courant'= courant \setminus \{v?\}

\end{schema}

\begin{schema}{est_presente}
\Xi parking \\
 v?: voiture \\
 presente!: \nat

\where

v? \in courant \implies presente!=1 \\
v? \notin courant \implies presente!=0

\end{schema}


\begin{schema}{NB_place}
\Xi parking \\
  NB!: \nat
\where
  NB!= capacite-(#courant)
\end{schema}
```

1 pts

1 pts

Exercice 03 :

```
chan approcheA = [1] of { bit };
chan approcheB = [1] of { bit };

chan feu_vert_A  = [1] of { bit };
chan feu_rouge_A = [1] of { bit };

chan feu_vert_B  = [1] of { bit };
chan feu_rouge_B = [1] of { bit };

int nb_vert= 0;


/* ---------------- FEU ROUTE A ---------------- */

proctype FeuRouteA()
{


    do
    :: approcheA ? 1 ->
        feu_vert_A ? 1;      /* attente feu vert */
        nb_vert++;

       printf("Voiture passe sur Route A\n");
       assert(nb_vert== 1);
```

0.25 pts

0.25 pts

0.25 pts

0.25 pts

0.25 pts

0.25 pts

1 pts

0.25 pts

```
        feu_rouge_A ! 1;      /* retour au rouge */
        nb_vert--;
    od
}

/* ---------------- FEU ROUTE B ---------------- */

proctype FeuRouteB()
{


    do
    :: approcheB ? 1 ->
        feu_vert_B ? 1;

        nb_vert++;

        printf("Voiture passe sur Route B\n");

      assert(nb_vert== 1);

        feu_rouge_B ! 1;
        nb_vert--;
    od
}


/* ------------- CONTROLEUR DES FEUX ------------- */

proctype Controleur()
{
    do
    :: atomic {
        /* Autoriser Route A */
        feu_vert_A ! 1;
        feu_rouge_B ? 1;      /* Route B attend rouge */
    }

    :: atomic {
        /* Autoriser Route B */
        feu_vert_B ! 1;
        feu_rouge_A ? 1;
    }
    od
}


/* -------------------- INIT -------------------- */

init
{
    run Controleur();
    run FeuRouteA();
    run FeuRouteB();

    /* génération de voitures */
    do
    :: approcheA ! 1
```

1 pts

1 pts

0.25 pts

1 pts

```
    :: approcheB ! 1
    od
}
```

Question 02 :

Exercice 04 :

Theorem plus_com : forall a b : nat, a + b = b + a. 1 pts

Proof.

induction a as [| a' IH]. 0.5 pts

- (* Cas a = 0 *)

intros b. simpl. 0.25 pts

(* on doit montrer : b = b + 0 *)

induction b as [| b' IHb]. 0.5 pts

+ simpl. reflexivity. 0.25 pts

+ simpl. rewrite <- IHb. reflexivity. 0.5 pts

- (* Cas a = S a' *)

intros b. simpl. 0.25 pts

(* On veut : S (a' + b) = b + S a' *)

rewrite IH. 0.5 pts

(* Reste à montrer : b + S a' = S (b + a') *)

induction b as [| b' IHb]. 0.5 pts

+ simpl. reflexivity. 0.25 pts

+ simpl. rewrite IHb. reflexivity. 0.5 pts

Qed.