

<

Cours Modélisation et vérification des systèmes informatiques

Exercices

Examen écrit et TP Noté du 27 novembre 2025

par Dominique Méry

4 décembre 2025

Exercice 1 Soit le contrat suivant :

```

variables int x, int y, int z
requires P(x0, y0, z0)
ensures Q(x0, y0, z0, xf, yf, zf)
begin
  0 : x = 10 ∧ y = z+x ∧ z = 2·x
  y := z+x
  1 : x = 10 ∧ y = x+2·10
end

```

Question 1.1 Compléter le module `ex25_1.tla` en définissant la précondition et la postcondition.**Question 1.2** Compléter l'action `step0_1` et vérifier que le module est sans blocage.**Question 1.3** Compléter le module en vérifiant que le contrat est correct c'est-à-dire que la propriété de correction partielle est satisfaite ou pas.

Listing 1 – labelex251

```

----- MODULE ex25_1 -----
EXTENDS Integers , TLC
CONSTANTS x0,y0,z0 (* x0,y0,z0 are the initial values *), un
P(u,v,w) == u=10 /\ v=u+w /\ w=2*u
Q(u0,v0,w0,u,v,w) ==

-----
ASSUME P(x0,y0,z0)

-----
VARIABLES x,y,z,pc

step0_1 == pc="0" /\ y '=_z+x_/_pc ='1" /\ x '=_x/_z '=z

skip == UNCHANGED <<x,y,z,pc>>

-----
Init ==
Next == step0_1 \/ skip

-----
safety pc == pc="1" =>
=====
```

Exercice 2 Soit le contrat suivant :

```

variables int x,int y,int z
requires P(x0,y0,z0)
ensures Q(x0,y0,z0,xf,yf,zf)
begin
  0 : x = 1  $\wedge$  y = 3  $\wedge$  x+y = 12
  x := y+x
  1 : x = 567  $\wedge$  y = 34
end

```

Question 2.1 Compléter le module `ex25_2.tla` en définissant la précondition et la postcondition.

Question 2.2 Compléter l'action `step0_1` et vérifier que le module est sans blocage.

Question 2.3 Compléter le module en vérifiant que le contrat est correct c'est-à-dire que la propriété de correction partielle est satisfaite ou pas.

Listing 2 – labelex251

```

----- MODULE ex25_2 -----
EXTENDS Integers , TLC
CONSTANTS x0,y0 (* x0,y0 are the initial values *), un
pre(u,v) ==
post(u0,v0,u,v) ==

----- ASSUME pre(x0,y0) -----
----- VARIABLES x,y,pc -----
step0_1 ==
skip == UNCHANGED <<x,y,pc>>
----- Init ==
Next == step0_1 \ / skip
----- safetypc ==
=====
```

Exercice 3 Soit le programme C suivant :

Listing 3 – summation.c

```

#include <stdio.h>

int ffS(int n) {
    int ps = 0;
    int k = 0;
    int ok=k, ops = 0;
    while (k < n) {
        ok=k;ops=ps;
        k = ok + 1;
        ps = ops + k;
    };
}
```

```

    return ps;
}
int fS(int n) {
    int ps = 0;
    int k = 0;
    while (k < n) {
        ps = ps + k+1;
        k = k + 1;
    };
    return ps;
}

int main()
{
    for (int i = 0; i < 11; ++i){
        printf("Value_for_z=%d_is_%d\n", i ,fS(i));
    }
    return 0;
}

```

Question 3.1 Ecrire les données en entrée de ce programme et ce qui est calculer.

Question 3.2 Donner la liste des variables utiles pour exprimer le calcul à partir des valeurs.

Question 3.3 Soit le code suivant extrait du programme ci-dessus :

Listing 4 – labelsummation.c

```

int ps = 0;
int k = 0;
int r;
inloop: while (k < x) {
    k = k + 1;

    ps = ps + k+1;
};

outloop: r = ps ;

```

Traduire ce code sous la forme d'actions suivant les étiquettes inloop et outloop.

Question 3.4 Vérifier la correction partielle et l'absence des erreurs à l'exécution.

Exercice 4 Soit le programme suivant :

Listing 5 – average.c

```

int a(int n,int m) {
    int r;
    r = (m+n) / 2;
    return r;
}

```

Question 4.1 Ecrire une relation entre les valeurs initiales n_0 , m_0 et r_0 et les valeurs finales de n_f , m_f et r_f des variables n, m, r . Traduire cette relation dans un module TLA⁺.

Question 4.2 L'exécution du programme *summation.c* conduit à une valeur inattendue pour certaines valeurs de *n0* et de *m0*. Analyser cette question avec l'outil *TLA⁺*.

Exercice 5 Soient les deux annotations suivantes. Pour chacune de ces annotations, traduire les annotations en unb module *TLA⁺* et vérifier ou non la correction de cette annotation.

Question 5.1

On suppose que p est un nombre premier :

```
ℓ₁ : x = 2p ∧ y = 2p+1 ∧ x·y = 22·p+1
x := y+x+2x
ℓ₂ : x = 5·2p ∧ y = 2p+1
```

Question 5.2

```
ℓ₁ : x = 1 ∧ y = 12
x := 2·y
ℓ₂ : x = 1 ∧ y = 24
```