

Cours MOdélisation, Vérification et EXpérimentations  
Exercices  
Série A Annotation, modélisation, vérification, validation en TLA<sup>+</sup> et PlusCal avec l'outil  
TLC  
par Dominique Méry  
24 janvier 2026

## Table des matières

<b>1 TD1</b>	<b>2</b>
<b>2 TD2</b>	<b>3</b>
<b>3 TD3</b>	<b>9</b>

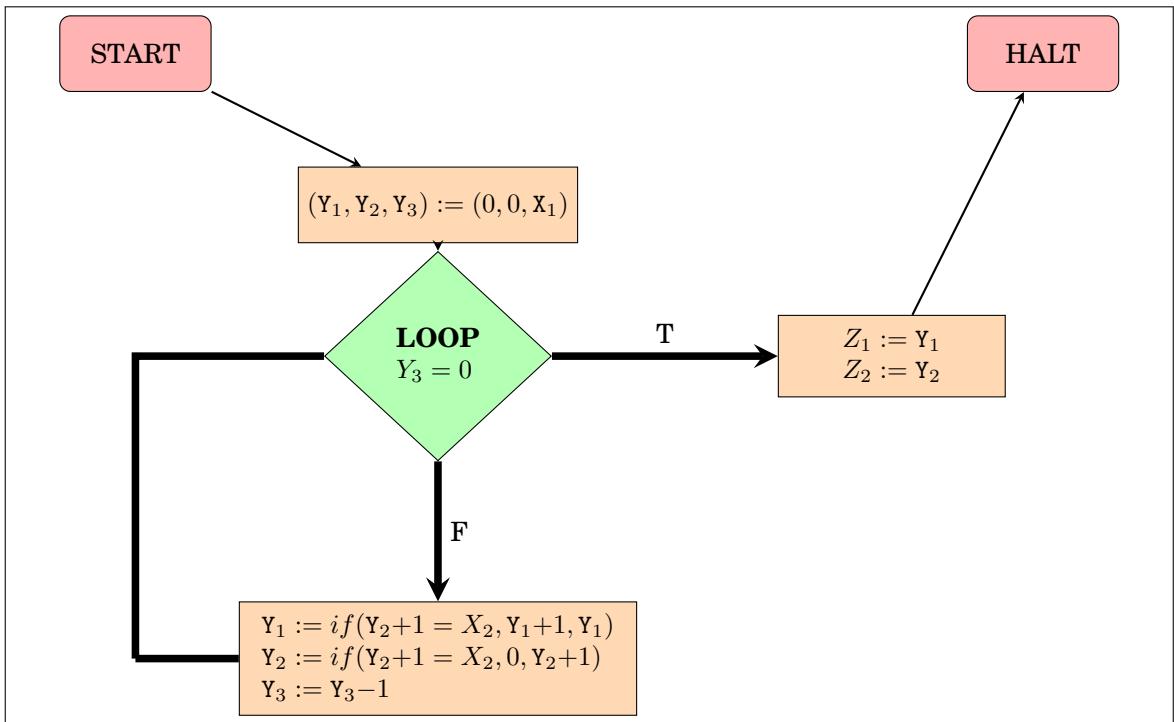


FIGURE 1 – Organigramme de calcul de la division entière

## 1 TD1

### Exercice 1 (malgtd1ex1)

Le PGCD de deux nombres vérifie les propriétés suivantes :

- $\forall a, b \in \mathbb{N}. \text{pgcd}(a, b) = \text{pgcd}(b, a)$
- $\forall a, b \in \mathbb{N}. \text{pgcd}(a, a+b) = \text{pgcd}(a, b)$

*mov*

- Ecrire une spécification TLA<sup>+</sup> calculant le PGCD de deux nombres donnés.
- Donner une explication ou une justification de la correction de cette solution

### Exercice 2 (malgtd1ex2)

L'accès à une salle est contrôlé par un système permettant d'observer les personnes qui entrent ou qui sortent de cette salle. Ce système est un ensemble de capteurs permettant d'identifier le passage d'une personne de l'extérieur vers l'intérieur et de l'intérieur à l'extérieur. Le système doit garantir qu'au plus  $\max$  personnes soient dans la salle. Ecrire un module TLA<sup>+</sup> permettant de modéliser un tel système respectant la propriété attendue.

### Exercice 3 (malgtd1ex3)

On considère l'algorithme suivant décrit par un organigramme ou flowchart de la figure 1. Cet algorithme calcule le reste et le quotient de la division de  $x_1$  par  $x_2$  :  $0 \leq z_2 \leq x_2 \wedge x_1 = z_1 \cdot x_2 + z_2$ . On suppose que  $x_1$  et  $x_2$  sont positifs et non nuls.

**Question 3.1** Donner la précondition et la postcondition associées à cet algorithme.

**Question 3.2** Traduire cet algorithme sous forme d'un module TLA<sup>+</sup>.

**Question 3.3** Tester les valeurs des variables à l'exécution.

**Question 3.4** Montrer que cet algorithme est partiellement correct par rapport à sa précondition et à sa postcondition qu'il faudra énoncer.

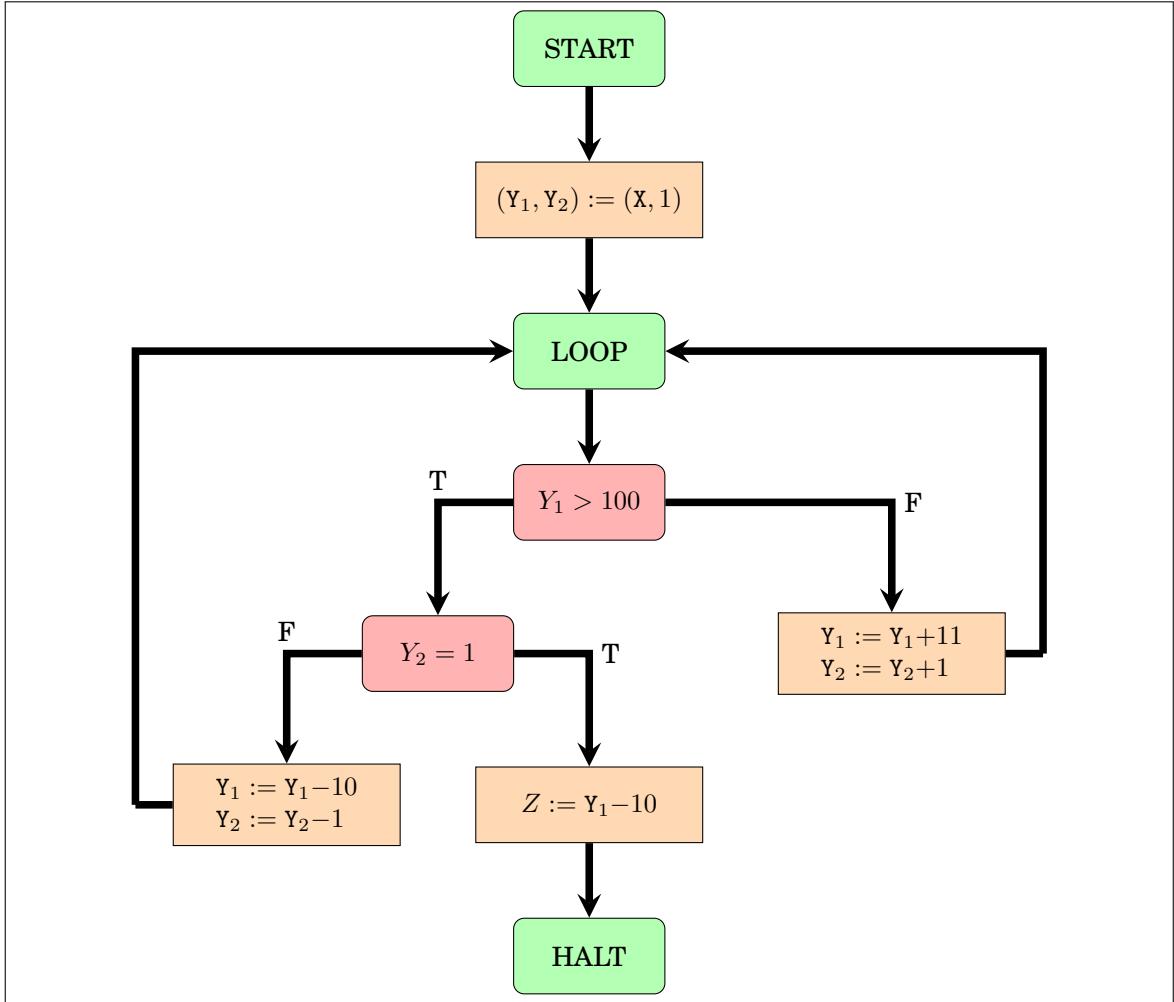


FIGURE 2 – Flowchart du calcul de la fonction de McCarthy

## 2 TD2

### Exercice 4 (malgtd1ex4)

La fonction de McCarthy  $f91$  est définie pour tout entier  $x$   $f91(x) = \text{if } x > 100 \text{ then } x-10 \text{ else } 91 \text{ fi.}$

**Question 4.1** Définir le contrat établissant la correction partielle de l'algorithme ALG91 de la figure 2 qui est réputé calculer la fonction  $f91$

**Question 4.2** Construire un module  $TLA^+$  modélisant les différents pas de calcul.

**Question 4.3** Evaluer l'algorithme en posant des questions de sûreté suivantes :

1. l'algorithme est partiellement correct.
2. l'algorithme n'a pas d'erreurs à l'exécution.

### Exercice 5 (malgtd1ex5, malgtd1ex5bis)

Soit le schéma de la figure 3 définissant un calcul déterminant, si un nombre entier naturel est premier ou non.

**Question 5.1** Ecrire un module  $TLA/TLA^+$  modélisant ce schéma de calcul et montrer que le modèle est sans blocage.

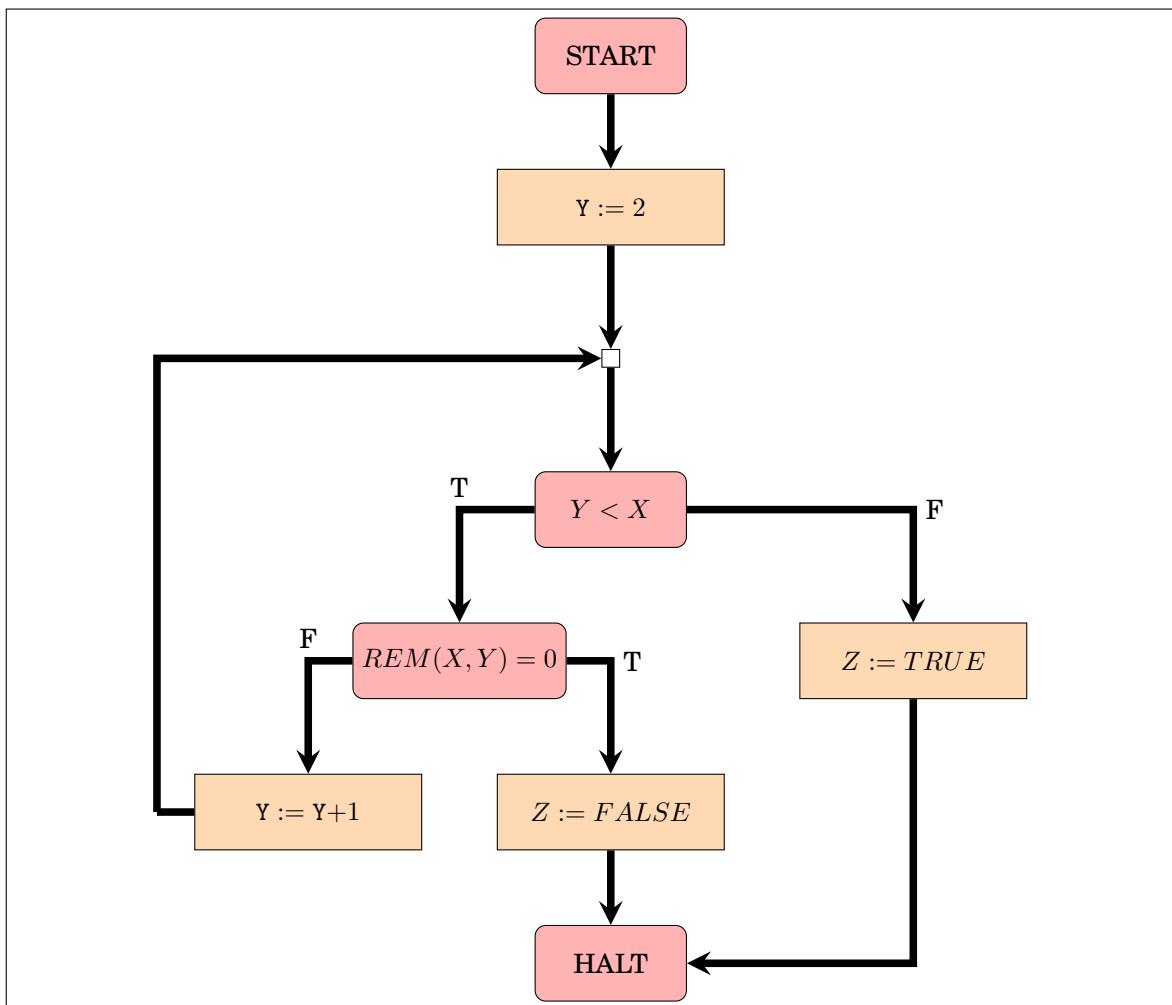


FIGURE 3 – Flowchart pour le test de primalité

**Question 5.2** Définir la propriété  $\text{prime}(x)$  qui est vraie si  $x$  est premier et faux sinon.

**Question 5.3** Ecrire le contrat présumé du calcul du flowchart de la figure 3

**Question 5.4** Enoncer et vérifier la correction partielle

**Question 5.5** Enoncer et vérifier l'absence d'erreurs à l'exécution.

**Exercice 6** (*malgtd1ex11,pluscal\_max.tla*)

```

Variables : X,Y,Z
Requires :  $x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}$ 
Ensures :  $z_f = \max(x_0, y_0)$ 

 $\ell_0 : \{ \dots \}$ 
if  $X < Y$  then
   $\ell_1 : \{ \dots \}$ 
   $Z := Y;$ 
   $\ell_2 : \{ \dots \}$ 
else
   $\ell_3 : \{ \dots \}$ 
   $Z := X;$ 
   $\ell_4 : \{ \dots \}$ 
;
 $\ell_5 : \{ \dots \}$ 
```

**Algorithme 1:** maximum de deux nombres non annotée

**Question 6.1** Ecrire un module  $TLA^+$  qui traduit la relation de transition de cet algorithme (Algorithme 6) selon les instructions. Pour cela, vous utiliserez la fonctionnalité offerte par la traduction d'un algorithme PlusCal en  $TLA^+$ . La figure est intitulée maximum de deux nombres non annotée.

**Question 6.2** Compléter l'algorithme intitulé maximum de deux nombres non annotée en l'annotant.

**Question 6.3** Vérifier que l'annotation est correcte.

**Question 6.4** Enoncer et vérifier la correction partielle et montrer que le contrat de correction partielle est satisfait.

**Question 6.5** Compléter le module  $TLA^+$  en définissant l'invariant construit avec les annotations et vérifier le contrat.

**Exercice 7** 

Nous allons utiliser la fonctionnalité de traduction d'un algorithme PlusCal en un module  $TLA$  pour vérifier des algorithmes. Pour chaque question, on écrira un module  $TLA$  contenant une expression de l'algorithme puis on traduira par un module et ensuite on analysera le module obtenu par rapport à la correction partielle et l'absence d'erreurs à l'exécution. Cet exercice ressemble aux exercices précédents mais se focalise sur le langage algorithmique PlusCal qui est traduit automatiquement comme cela a été fait manuellement.

```

Variables : X,Y,Z
Requires :  $x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}$ 
Ensures :  $z_f = \max(x_0, y_0)$ 

 $\ell_0 : \{x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 
if  $X < Y$  then
     $\ell_1 : \{x < y \wedge x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 
     $Z := Y;$ 
     $\ell_2 : \{x < y \wedge x = x_0 \wedge y = y_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z} \wedge z = y_0\}$ 
else
     $\ell_3 : \{x \geq y \wedge x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 
     $Z := X;$ 
     $\ell_4 : \{x \geq y \wedge x = x_0 \wedge y = y_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z} \wedge z = x_0\}$ 
;
 $\ell_5 : \{z = \max(x_0, y_0) \wedge x = x_0 \wedge y = y_0 \wedge x_0, y_0 \in \mathbb{N} \wedge z_0 \in \mathbb{Z}\}$ 

```

### Algorithme 2: maximum de deux nombres non annotée

#### Question 7.1 (appex4\_1\_1)

Soit l'annotation suivante :

```

 $\ell_1 : x = 10 \wedge y = z+x \wedge z = 2 \cdot x$ 
 $y := z+x$ 
 $\ell_2 : x = 10 \wedge y = x+2 \cdot 10$ 

```

Traduire en PlusCal.

#### Question 7.2 (appex4\_1\_2)

On suppose que  $p$  est un nombre premier :

```

 $\ell_1 : x = 2^p \wedge y = 2^{p+1} \wedge x \cdot y = 2^{2 \cdot p + 1}$ 
 $x := y + x + 2^x$ 
 $\ell_2 : x = 5 \cdot 2^p \wedge y = 2^{p+1}$ 

```

#### Question 7.3 (appex4\_1\_3)

```

 $\ell_1 : x = 1 \wedge y = 12$ 
 $x := 2 \cdot y$ 
 $\ell_2 : x = 1 \wedge y = 24$ 

```

#### Question 7.4 (appex4\_1\_4)

```

 $\ell_1 : x = 11 \wedge y = 13$ 
 $z := x; x := y; y := z;$ 
 $\ell_2 : x = 26/2 \wedge y = 33/3$ 

```

#### Exercice 8 (tla\_squareroot, pluscal\_squareroot)

On considère l'algorithme squareroot calculant la racine carrée entière d'un nombre naturel  $x \in \mathbb{N}$ .

*VARIABLES*  $X, Y1, Y2, Y3, Z$

---

$pre(x0, y10, y20, y30, z0) \stackrel{def}{=} U$   
 $U \stackrel{def}{=} ( X, Y1, Y2, Y3, Z )$   
 $u0 \stackrel{def}{=} ( x0, y10, y20, y30, z0 )$   
 $post(x0, y10, y20, y30, z0, xf, y1f, y2f, y3f, zf) \stackrel{def}{=}$

---

*REQUIRES*  $pre(x0, y10, y20, y30, z0)$   
*ENSURES*  $post(x0, y10, y20, y30, z0, xf, y1f, y2f, y3f, zf)$

---

$\ell_0 : pre(u0) \wedge u = u0$   
 $(Y1, Y2, Y3) := (0, 1, 1)$   
 $\ell_1 : pre(u0) \wedge x = x0 \wedge z = z0 \wedge y2 = (y1+1) \cdot (y1+1) \wedge y3 = 2 \cdot y1 + 1 \wedge y1 \cdot y1 \leq x$   
**WHILE**  $Y2 \leq X$  **DO**  
 $\ell_2 :$   
 $(Y1, Y2, Y3) := (Y1+1, Y2+Y3+2, Y3+2);$   
 $\ell_3 : OD;$   
 $\ell_4 :$   
 $Z := Y1;$   
 $\ell_5 :$

Listing 1 – mainsquareroot.c

```
#include <stdio.h>
#include <limits.h>
int spower2(int x)
{int z,y1,y2,y3,r;
r=0;y1=0;y2=1;y3=1;
while (y2<= x)
{
    y1=y1+1;
    y2= y2 + y3 + 2;
    y3=y3+2;

}
r=y1;
return(r);
}

int main ()
{
int counter;
for( counter=0; counter<5; counter++ )
{
int v,r;
printf("Enter_a_natural_number:");
scanf("%d", &v);
r = spower2(v);
printf ("Power_2:_%d---->_%d\n", v,r);

};
}
```

**Question 8.1** Définir les deux assertions  $pre$  et  $post$  qui établissent le contrat de cet algorithme.

**Question 8.2** Annoter l'algorithme et le traduire en PlusCal. On pourra faire deux types d'annotations :

- soit dans les instructions `assert`.
- soit sous la forme d'un invariant global.

**Question 8.3** Vérifier la correction partielle et l'absence d'erreurs à l'exécution de cet algorithme en utilisant la traduction en TLA<sup>+</sup>.

### 3 TD3

**Exercice 9** (*malgtd1ex10, malgtd1ex10bis, malgtd1ex10ter, malgtd1ex10last*)

Pour montrer que chaque annotation est correcte ou incorrecte, on propose de procéder comme suit :

- Traduire cette annotation sous la forme d'un contrat.
- Vérifier les conditions de vérification du contrat

$\ell_1 : P_{\ell_1}(v)$ $v := f(v, c)$ $\ell_2 : P_{\ell_2}(v)$
--

variables v requires $pre(v_0)$ ensures $post(v_0, v_f)$ begin $\ell_1 : Q_1(v_0, v)$ $v := f(v, c)$ $\ell_2 : Q_2(v_0, v)$ end
--

On rappelle qu'un contrat est valide si les trois conditions suivantes sont valides :

- (init)  $pre(v_0) \wedge v = v_0 \Rightarrow Q_1(v_0, v)$
- (concl)  $pre(v_0) \wedge Q_2(v_0, v) \Rightarrow post(v_0, v)$
- (induct)  $pre(v_0) \wedge Q_1(v_0, v) \wedge cond_{\ell_1, \ell_2}(v) \wedge v' = f(v, c) \Rightarrow Q_2(v_0, v')$

Les deux propriétés (init) et (concl) sont valides par construction et la seule propriété à montrer correcte ou incorrecte est la propriété (induct).

**Question 9.1** (*malgtd1ex10*)

$\ell_1 : x = 3 \wedge y = z+x \wedge z = 2 \cdot x$ $y := z+x$ $\ell_2 : x = 3 \wedge y = x+6$
---

**Question 9.2** (*malgtd1ex10bis*)

Pour les deux exemples qui suivent, on considère des cas et on doit donner une interprétation.

$\ell_1 : x = 2^4 \wedge y = 2 \wedge x \cdot y = 2^6$ $x := y+x+2^x$ $\ell_2 : x = 2^{10} \wedge y = 2$
--

$\ell_1 : x = 2^4 \wedge y = 2 \wedge x \cdot y = 2^5$ $x := y+x+2^x$ $\ell_2 : x = 2^{10} \wedge y = 2$
--

**Question 9.3** (*malgtd1ex10ter.tla*)

$\ell_1 : x = 1 \wedge y = 12$ $x := 2 \cdot y + x$ $\ell_2 : x = 1 \wedge y = 25$
--

**Question 9.4** (*malgtd1ex10last.tla*)

$\ell_1 : x = 11 \wedge y = 13$   
 $z := x; x := y; y := z;$   
 $\ell_2 : x = 26/2 \wedge y = 33/3$

**Exercice 10** Montrer que chaque annotation est correcte ou incorrecte selon les conditions de vérifications énoncées comme suit

$$\forall x, y, , x', y'. P_\ell(x, y) \wedge cond_{\ell, \ell'}(x, y) \wedge (x', y') = f_{\ell, \ell'}(x, y) \Rightarrow P_{\ell'}(x', y')$$

$\ell_1 : x = 10 \wedge y = z+x \wedge z = 2 \cdot x$ $y := z+x$ $\ell_2 : x = 10 \wedge y = x+2 \cdot 10$	$\ell_1 : x = 1 \wedge y = 12$ $x := 2 \cdot y$ $\ell_2 : x = 1 \wedge y = 24$
$—$ $On \quad suppose \quad que \quad p \quad est$ $un \quad nombre \quad premier \quad :$ $\ell_1 : x = 2^p \wedge y = 2^{p+1} \wedge x \cdot y = 2^{2 \cdot p+1}$ $x := y+x+2^x$ $annol_2 : x = 5 \cdot 2^p \wedge y = 2^{p+1}$	$—$ $\ell_1 : x = 11 \wedge y = 13$ $z := x; x := y; y := z;$ $\ell_2 : x = 26/2 \wedge y = 33/3$

On rappelle qu'un contrat pour la correction partielle d'un petit programme est donné par les éléments ci-dessous en colonne de gauche et que les conditions de vérification associées sont définies par le texte de la colonne de droite.

---

#### Contrat de la correction partielle

variables type $X$ <hr/> definitions $def1 \stackrel{\text{def}}{=} \text{text1}$ <hr/> requires $pre(x_0)$ <hr/> ensures $post(x_0, x_f)$ <hr/> begin $0 : P_0(x_0, x)$ instruction <sub>0</sub> $1 : P_i(x_0, x)$ instruction <sub>1</sub> $f : P_f(x_0, x)$ end	Conditions de vérification $— pre(x_0) \wedge x = x_0 \Rightarrow P_0(x_0, x)$ $— pre(x_0) \wedge P_f(x_0, x) \Rightarrow post(x_0, x)$ $—$ Pour toutes les paires $\ell, \ell'$ , telles que $\ell \rightarrow \ell'$ , on vérifie que, pour toutes les valeurs $x, x' \in \text{MEMORY}$ $\left( \begin{array}{l} (pre(x_0) \wedge P_\ell(x_0, x)) \\ \wedge cond_{\ell, \ell'}(x) \wedge x' = f_{\ell, \ell'}(x) \end{array} \right) \Rightarrow P_{\ell'}(x_0, x')$
---	---

#### Exercice 11 (prog23-4.c)

Soit le contrat suivant qui met en jeu les variables  $X, Y, Z, C, R$ .

**VARIABLES** int  $X, Y, Z, C, R$

**REQUIRES**  $x_0, y_0, z_0, c_0, r_0 \in \mathbb{Z}$

**ENSURES**  $r_f = 0$

**BEGIN**

$0 : x = x_0 \wedge y = y_0 \wedge z = z_0 \wedge c = c_0 \wedge r = r_0 \wedge x_0, y_0, z_0, c_0, r_0 \in \mathbb{Z}$

$(X, Z, Y) := (49, 2 \cdot C, (2 \cdot C + 1) \cdot (2 \cdot C + 1));$

$1 : x = 49 \wedge z = 2 \cdot c \wedge y = (z + 1) \cdot (z + 1)$

$Y := X + Z + 1;$

$2 : x = 49 \wedge z = 2 \cdot c \wedge y = (c + 1) \cdot (c + 1)$

**END**

**Question 11.1** Ecrire les conditions de vérification associée au contrat ci-dessus en vous aidant du rappel de la définition de ces conditions de vérification.

**Question 11.2** Simplifier les conditions de vérification et préciser les conditions que doivent vérifier les valeurs initiales des variables  $X, Y, Z, C, R$  pour que les conditions de vérification soient toutes vraies. En particulier, il faudra s'assurer que la pr<sup>é</sup>condition est satisfaisable.

**Exercice 12 ()**

On considère le petit programme se trouvant à droite de cette colonne. Nous allons poser quelques questions visant à compléter les parties marquées en gras et visant à définir la relation de calcul.

On notera  $\text{pre}(n_0, x_0, b_0)$  l'expression  $n_0, x_0, b_0 \in \mathbb{Z}$  et  $\text{in}(n, b, n_0, x_0, b_0)$  l'expression  $n = n_0 \wedge b = b_0 \wedge \text{pre}(n_0, x_0, b_0)$

**Question 12.1** Donner l'assertion Requ<sup>ires</sup> en complétant ce qui est déjà mentionné et en reportant le texte complet de cette assertion Requ<sup>ires</sup> dans votre copie.

On rappelle que la relation de transition de  $\ell$  vers  $\ell'$ , notée  $a(\ell, \ell')$ , est définie par une relation de la forme  $\text{cond}_{\ell, \ell'}(v) \wedge v' = f_{\ell, \ell'}(v)$ .

**Question 12.2** Ecrire les relations de transition entre les étiquettes successives :  $a(\ell_0, \ell_1), a(\ell_1, \ell_2), a(\ell_2, \ell_3), a(\ell_3, \ell_6), a(\ell_1, \ell_4), a(\ell_4, \ell_5), a(\ell_5, \ell_6)$ .

**VARIABLES** int  $N, X, B$

**REQUIRES**  $n_0, x_0, b_0 \in \mathbb{Z}$

**ENSURES**  $\begin{cases} n_0 < b_0 \Rightarrow x_f = \text{question1} \\ n_0 \geq b_0 \Rightarrow x_f = \text{question1} \\ n_f = n_0 \wedge b_f = b_0 \end{cases}$

**BEGIN**

$\ell_0 :$

$X := N;$

$\ell_1 :$

**IF**  $X < B$  **THEN**

$\ell_2 :$

$X := X \cdot X + 2 \cdot B \cdot X + B \cdot B;$

$\ell_3 :$

**ELSE**

$\ell_4 :$

$X := B;$

$\ell_5 :$

**FI**

$\ell_6 :$

**END**

**Fin de la série AA**