Génie Logiciel Avancé

Année 2016-2017

Delphine Longuet, Thibaut Balabonski, Robin Pelle, Hadi Zaatiti
longuet@lri.fr, blsk@lri.fr, pelle@lri.fr, hadizaatiti@gmail.com

Examen

14 décembre 2016
Durée : 2 h 30

Seuls les transparents du cours (éventuellement annotés) sont autorisés.

Exercice 1 (barème indicatif : 11 points)
La fonction suivante calcule la multiplication de deux nombres entiers a et b avec b positif ou nul.

```
int mult(int a, int b) {
    int res = 0;
    while(b > 0) {
        if(b mod 2 != 0) {
            res = res + a;
            b = b - 1;
        }
        a = 2 * a;
        b = b/2;
    }
    return res;
}
```

1. Uniquement à partir de la spécification, donner exactement 3 tests pour la fonction `mult`. Pour chacun des tests, donner son objectif, des données d'entrée concrètes et le résultat attendu.

Partie 1. Critères sur le graphe de flot de contrôle

2. Donner le graphe de flot de contrôle de la fonction `mult`.
3. Donner une expression régulière représentant l'ensemble des chemins du graphe.

Toutes les instructions.

4. Donner le chemin le plus court permettant de satisfaire le critère « toutes les instructions ». On notera ce chemin `ch1`.
5. Par exécution symbolique, calculer la condition associée à ce chemin. Ce chemin est-il faisable ? Si oui, donner un test concret pour ce chemin, sinon, expliquer pourquoi.

Toutes les décisions. On considère l'ensemble de chemins $T = \{\texttt{ch1}\}$.

6. Compléter T par le plus court chemin permettant à T de satisfaire le critère « toutes les décisions »(également appelé « toutes les branches » ou « tous les arcs »). Justifier que le critère est satisfait.
7. Par exécution symbolique, calculer la condition associée à ce nouveau chemin. Est-il faisable ? Expliquer.
8. Déduire de la question précédente une condition suffisante pour qu'un chemin du graphe de flot de contrôle de cette fonction soit infaisable.
9. Ajouter finalement à T le plus court chemin faisable permettant à T de satisfaire le critère « toutes les décisions ». On notera ce chemin `ch2`.
10. Calculer par exécution symbolique la condition associée à ce chemin. Donner un test concret pour ce chemin.

Partie 2. Critères sur le graphe de flot de données

11. Annoter le graphe de flot de contrôle donné en question 2 par le flot de données de `mult`, c'est-à-dire les définitions et utilisations des variables.
12. Pour chacune des variables suivantes, donner une expression régulière représentant le flot de données pour cette variable. Préciser si le flot de données présente des anomalies ou non. En cas d'anomalie, expliquer s'il s'agit d'une erreur et pourquoi.
    (a) la variable b ;
    (b) la variable a.

Toutes les définitions.

13. Pour chacune des variables suivantes, donner un ensemble de chemins faisables permettant de satisfaire le critère « toutes les définitions ». Justifier soigneusement que le critère est satisfait. Remarque : il est possible d'utiliser les chemins de la partie 1.
    (a) la variable b ;
    (b) la variable a.

Toutes les utilisations.

14. Pour chacune des variables suivantes, donner un ensemble de chemins faisables permettant de satisfaire le critère « toutes les utilisations ». Justifier soigneusement que le critère est satisfait, ou expliquer pourquoi il ne peut pas l'être le cas échéant.
    (a) la variable res ;
    (b) la variable b.

Exercice 2 (barème indicatif : 6 points)
On veut prouver que le programme suivant effectue la multiplication de a par b pour $b \geq 0$.

```
x := 0;
y := 0;
WHILE y < b DO
   x := x + a;
   y := y + 1
```

Les règles du calcul de Hoare sont rappelées à la fin de l'énoncé. Dans vos réponses aux questions, **l'application d'une règle doit toujours être justifiée.**

1. Écrire la spécification du programme sous forme de pré et post-conditions.
2. Quel est le triplet de Hoare à prouver ?
3. Trouver un invariant Inv pour la boucle `WHILE`.

Les questions suivantes vous guident dans la preuve de la boucle `WHILE`, en partant des feuilles de l'arbre de preuve. Elles peuvent cependant être traitées indépendamment les unes des autres.

4. Dériver le triplet de Hoare :

$$\{x = a \times y \wedge y < b\}\ \texttt{x := x + a; y := y + 1}\ \{x = a \times y \wedge y \leq b\}$$

Indication : trouver une propriété S qui permette de démontrer le triplet

$$\{S\}\ \texttt{y := y + 1}\ \{x = a \times y \wedge y \leq b\}$$

5. On note `Loop` la boucle `WHILE` du programme. Compléter la preuve précédente afin de dériver le triplet :

$$\{x = a \times y \wedge y \leq b\}\ \texttt{Loop}\ \{x = a \times y \wedge y = b\}$$

6. Terminer la preuve de la boucle `WHILE` en complétant votre preuve de manière à dériver le triplet :

$$\{b \geq 0 \wedge x = 0 \wedge y = 0\}\ \texttt{Loop}\ \{x = a \times b\}$$

7. Donner un variant pour la boucle de ce programme, c'est-à-dire une expression toujours positive ou nulle et qui décroît strictement à chaque tour de boucle.

Exercice 3 (barème indicatif : 3 points)
On considère le programme suivant qui renvoie *true* si et seulement si l'entier v est présent dans le tableau t.

```
boolean contains(int t[], int v) {
    int i = 0;
    while(i < t.length) {
        if(t[i] == v) {
            return true;
        }
        i++;
    }
    return false;
}
```

1. Donner la spécification de ce programme en termes de pré et post-condition.
2. Donner un invariant pour la boucle.
   Indication : l'invariant est composé de deux propriétés, l'une portant sur le compteur de la boucle et l'autre portant sur la progression de la recherche de l'élément v dans le tableau. On rappelle que l'invariant, avec la négation de la condition du `while`, doit impliquer (une partie de) la post-condition.
3. Donner un variant pour la boucle, c'est-à-dire une expression toujours positive ou nulle et qui décroît strictement à chaque tour de boucle.

Calcul de Hoare

$$\frac{}{\{P\}\ \texttt{SKIP}\ \{P\}}\ \text{skip} \qquad \frac{}{\{P[x \mapsto exp]\}\ \texttt{x := exp}\ \{P\}}\ \text{affectation}$$

$$\frac{\{P \wedge cond\}\ \texttt{ins}_1\ \{Q\} \qquad \{P \wedge \neg cond\}\ \texttt{ins}_2\ \{Q\}}{\{P\}\ \texttt{IF cond THEN ins}_1\ \texttt{ELSE ins}_2\ \{Q\}}\ \text{ifthenelse}$$

$$\frac{\{P \wedge cond\}\ \texttt{ins}\ \{P\}}{\{P\}\ \texttt{WHILE cond DO ins}\ \{P \wedge \neg cond\}}\ \text{while}$$

$$\frac{P \Rightarrow P' \qquad \{P'\}\ \texttt{ins}\ \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\ \texttt{ins}\ \{Q\}}\ \text{consequence}$$

$$\frac{}{\{false\}\ \texttt{ins}\ \{P\}}\ \text{falseE} \qquad \frac{\{P\}\ \texttt{ins}_1\ \{Q\} \qquad \{Q\}\ \texttt{ins}_2\ \{R\}}{\{P\}\ \texttt{ins}_1\ \texttt{;}\ \texttt{ins}_2\ \{R\}}\ \text{sequence}$$

Solutions

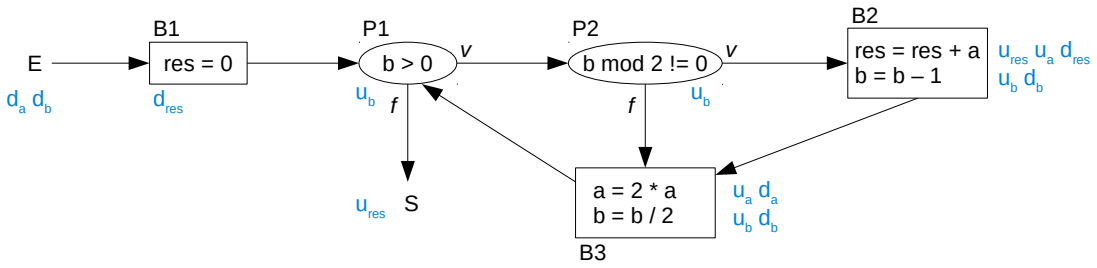
Exercice 1 (barème : 11,5 points)

1. (barème : 0,75)
   On demandait ici des tests fonctionnels (« uniquement à partir de la spécification »), donc où toutes les valeurs respectent les pré-conditions. En particulier, un test avec b négatif n'est pas un test fonctionnel, puisqu'on ne sait pas ce que la fonction est censée renvoyer dans ce cas.
   Les trois tests principaux pour cette fonction sont les suivants :
   - la multiplication par 0 avec $b = 0$ pour tester la limite de b ;
   - la multiplication par un entier négatif $a < 0$ (b étant nécessairement positif) pour tester que le résultat est bien négatif ;
   - la multiplication par un entier positif $a > 0$ pour tester que le résultat est positif.

   Comme d'habitude, il faut choisir des valeurs différentes entre les tests et éviter les cas trop particuliers comme $a = 1$ sauf si c'est l'objectif du test.

Objectif de test	Données d'entrée		Résultat attendu
	a	b	
$b = 0$	12	0	0
a négatif	-5	8	-40
a positif	7	9	63

2. (barème : 1)
   Graphe de flot de contrôle et de flot de données de la fonction.



3. (barème : 0,5)
   Expression régulière représentant l'ensemble des chemins du graphe :
   E B1 P1 (P2 (B2 B3 + B3) P1)* S
4. (barème : 0,5)
   Chemin le plus court pour satisfaire le critère « toutes les instructions » :
   E B1 P1 P2 B2 B3 P1 S (`ch1`)
   Il faut passer dans la boucle une fois en rendant la condition du if vraie.
5. (barème : 1)
   Exécution symbolique pour ce chemin.

	a	b	res	condition de chemin
E	a_0	b_0	res_0	true
B1	a_0	b_0	0	—
P1	a_0	b_0	0	(P1 vraie) $b_0 > 0$
P2	a_0	b_0	0	(P2 vraie) $b_0 \mod 2 \neq 0$
B2	a_0	$b_0 - 1$	a_0	—
B3	$2a_0$	$(b_0 - 1)/2$	a_0	—
P1	$2a_0$	$(b_0 - 1)/2$	a_0	(P1 fausse) $(b_0 - 1)/2 \leq 0$
S	$2a_0$	$(b_0 - 1)/2$	a_0	—

Condition de chemin finale : $b > 0$, b impair et $b \leq 1$. Donc $b = 1$.
Test pour ce chemin : $a = 6$, $b = 1$, résultat attendu : 6.

6. (barème : 0,5)
   On pose $T = \{\texttt{ch1}\}$, c'est-à-dire qu'on considère l'ensemble de chemins T contenant uniquement le chemin `ch1`. On cherche le chemin qu'il faut ajouter à l'ensemble T pour que cet ensemble satisfasse le critère « toutes les décisions ».
   Le plus court chemin permettant à T de satisfaire « toutes les décisions » est donc :
   E B1 P1 P2 B3 P1 S
   Il faut passer une fois dans la boucle en rendant la condition du if fausse.
   Justification de la satisfaction du critère « toutes les décisions » : P1 est vraie pour rentrer dans la boucle puis fausse pour en sortir, dans chacun de ces chemins ; P2 est vraie dans `ch1` et fausse dans ce deuxième chemin.
   Remarque. La question a été considérée comme ambiguë, par conséquent toutes les réponses proposant un chemin couvrant à lui seul toutes les décisions ont été acceptées. En particulier, le chemin E B1 P1 P2 B2 B3 P1 P2 B3 P1 S était une réponse possible, ainsi que le chemin E B1 P1 P2 B3 P1 P2 B2 B3 P1 S, même si ce n'était pas les chemins attendus.
7. (barème : 0,75)
   Exécution symbolique pour ce chemin.

	a	b	res	condition de chemin
E	a_0	b_0	res_0	true
B1	a_0	b_0	0	—
P1	a_0	b_0	0	(P1 vraie) $b_0 > 0$
P2	a_0	b_0	0	(P2 fausse) $b_0 \mod 2 = 0$
B3	$2a_0$	$b_0/2$	0	—
P1	$2a_0$	$b_0/2$	0	(P1 fausse) $b_0/2 \leq 0$
S	$2a_0$	$b_0/2$	0	—

   Condition de chemin finale : $b > 0$, b pair et $b \leq 0$. Condition insatisfiable donc ce chemin est infaisable. Puisque b est strictement positif, $b/2$ ne peut pas être égal à 0.
   Remarque. Suite à la question précédente, si l'exécution symbolique du chemin choisi est correcte, tous les points de la question ont été attribués.
8. (barème : 0,5)
   Il est impossible de sortir de la boucle après avoir rendu la condition du if fausse. En effet, il n'existe pas d'entier strictement positif tel que divisé par 2, il soit égal à 0. Tous les chemins qui finissent en P2 B3 P1 S sont donc infaisables.

9. (barème : 0,5)
   Plus court chemin faisable permettant à T de satisfaire « toutes les décisions » :
   E B1 P1 P2 B3 P1 P2 B2 B3 P1 S (`ch2`)
   On allonge le chemin précédent en passant une deuxième fois dans la boucle, en rendant la condition du if vraie, pour pouvoir sortir.
   Remarque. Suite à la question 6, si le chemin donné ici est différent de celui donné à la question 6 et couvre bien toutes les décisions, les points ont été attribués. La moitié des points a été donnée si le chemin couvre toutes les décisions mais n'est pas le plus court (il passe 3 fois dans la boucle par exemple).
10. (barème : 1)
    Exécution symbolique pour ce chemin.

	a	b	res	condition de chemin
E	a_0	b_0	res_0	true
B1	a_0	b_0	0	—
P1	a_0	b_0	0	(P1 vraie) $b_0 > 0$
P2	a_0	b_0	0	(P2 fausse) $b_0 \mod 2 = 0$
B3	$2a_0$	$b_0/2$	0	—
P1	$2a_0$	$b_0/2$	0	(P1 vraie) $b_0/2 > 0$
P2	$2a_0$	$b_0/2$	0	(P2 vraie) $b_0/2 \mod 2 \neq 0$
B2	$2a_0$	$b_0/2 - 1$	$2a_0$	—
B3	$4a_0$	$(b_0/2 - 1)/2$	$2a_0$	—
P1	$4a_0$	$(b_0/2 - 1)/2$	$2a_0$	(P1 fausse) $(b_0/2 - 1)/2 \leq 0$
S	$4a_0$	$(b_0/2 - 1)/2$	$2a_0$	—

    Condition de chemin finale : $b > 0$, b pair, $b/2 > 0$, $b/2$ impair et $b \leq 2$. Donc $b = 2$.
    Test pour ce chemin : $a = 7$, $b = 2$, résultat attendu : 14.
11. (barème : 1)
    (Voir le graphe donné en question 2.)
12. (barème : 1)
    (a) Flot de données pour b :
    $d_b u_b (u_b (u_b d_b u_b d_b + u_b d_b) u_b)^*$
    Toutes les définitions sont utilisées au moins une fois, dans tous les chemins du graphe, donc il n'y a pas d'anomalie du flot de données pour b.
    (b) Flot de données pour a :
    $d_a (u_a u_a d_a + u_a d_a)^*$
    La dernière définition de a dans B3 avant la sortie de la boucle n'est jamais utilisée, de même que la définition en B1 dans le chemin qui ne rentre pas dans la boucle. C'est une anomalie mais pas une erreur. La définition en B3 prépare le calcul pour le prochain tour de boucle, s'il a lieu, tandis que la définition en B1 permet de renvoyer 0 dans le cas où $b = 0$.
13. (barème : 1)
    Attention à la définition des critères sur le flot de données. Satisfaire le critère « toutes les définitions de b » ne signifie pas « passer par toutes les définitions de b ». Il faut trouver un ensemble de chemins tel que toutes les définitions de b soient *utilisées au moins une fois*.

(a) Toutes les définitions de b.
Les définitions de b sont en E, B2 et B3. Chacune de ces définitions est utilisée dans `ch1` : définition en E utilisée en P1, définition en B2 utilisée en B3, définition en B3 utilisée en P1. Donc `ch1` permet de couvrir toutes les définitions de b.

(b) Toutes les définitions de a.
Les définitions de a sont en E et B3. La définition en B3 n'est utilisée que si on passe deux fois dans la boucle, donc `ch2` permet de couvrir cette définition en l'utilisant en B2. Dans `ch2`, la définition en E est utilisée en B3. Donc `ch2` permet de couvrir toutes les définitions de a.

14. (barème : 1,5)
De la même manière ici, satisfaire le critère « toutes les utilisations de *res* » ne signifie pas « passer par toutes les utilisations de *res* ». Il faut trouver un ensemble de chemins qui couvre *tous les couples définition-utilisation* de *res*, pour que toutes les définitions soient utilisées de toutes les manières possibles.

(a) Toutes les utilisations de *res*.
Les définitions de *res* sont en B1 et en B2. Les utilisations sont en B2 et en S.

Déf	Util	chemin couvrant ce couple définition-utilisation
B1	B2	`ch1`
B1	S	E B1 P1 S
B2	B2	E B1 P1 P2 B2 B3 P1 P2 B2 B3 P1 S
B2	S	`ch1`

[Non demandé]
E B1 P1 S. Condition de chemin : $b = 0$.
E B1 P1 P2 B2 B3 P1 P2 B2 B3 P1 S. Condition de chemin : $b = 3$

(b) Toutes les utilisations de b.
Les définitions de b sont en E, B2 et B3. Les utilisations sont en P1, P2, B2 et B3.

Déf	Util	chemin couvrant ce couple définition-utilisation
E	P1	`ch1`
	P2	`ch1`
	B2	`ch1`
	B3	`ch2`
B2	P1	impossible
	P2	impossible
	B2	impossible
	B3	`ch1`
B3	P1	`ch1`
	P2	`ch2`
	B2	`ch2`
	B3	E B1 P1 P2 B3 P1 P2 B3 P1 P2 B2 B3 P1 S

Les couples (B2,P1), (B2,P2) et (B2,B2) sont impossibles à couvrir puisqu'il y a nécessairement une redéfinition de b en B3. Ca n'empêche pas de satisfaire le critère en couvrant tous les couples définition-utilisation possibles.

Le dernier chemin est le plus court chemin faisable couvrant le couple (B3,B3), il faut nécessairement passer une troisième fois dans la boucle pour pouvoir sortir. Le chemin qui rend la condition du if vraie au premier passage dans la boucle est aussi un chemin possible pour couvrir ce couple (et pas beaucoup plus long).
[Non demandé]
E B1 P1 P2 B3 P1 P2 B3 P1 P2 B2 B3 P1 S. Condition de chemin : $b = 4$.
ou bien E B1 P1 P2 B2 B3 P1 P2 B3 P1 P2 B2 B3 P1 S. Condition de chemin : $b = 5$.

Exercice 2 (barème : 5,5)

1. (barème : 1)
   La seule pré-condition est que b soit positif ou nul, a est quelconque. Les pré-conditions portent toujours uniquement sur les arguments de la fonction.
   La post-condition exprime le fait que la fonction calcule la multiplication de a par b dans la variable x. Plus proprement, on devrait écrire $result = a \times b$ puisque la post-condition ne doit porter que sur les arguments et la valeur renvoyée par la fonction.
   Pré-condition : $b \geq 0$
   Post-condition : $x = a \times b$
2. (barème : 0,5)
   Le triplet à prouver est $\{Pre\}$ `Prog` $\{Post\}$, c'est-à-dire

   $$\{b \geq 0\} \texttt{ x:=0; y:=0; WHILE y < b DO x:=x+a; y:=y+1 } \{x = a \times b\}$$

3. (barème : 1)
   La multiplication est calculée à chaque tour de boucle en ajoutant a à x et elle tend vers la valeur finale qui est $a \times b$. À chaque tour de boucle, on a fait y étapes du calcul final, donc $x = a \times y$. D'autre part, y est le compteur de la boucle et prend toutes les valeurs entre 0 et b inclus, puisque $y = b$ au moment où on sort de la boucle.
   Invariant : $x = a \times y \wedge y \leq b$.
4. (barème : 1)
   On applique la règle de séquence pour découper la preuve en deux.

   $$\frac{\{x = a \times y \wedge y < b\} \texttt{ x := x+a } \{S\} \qquad \{S\} \texttt{ y := y+1 } \{x = a \times y \wedge y \leq b\}}{\{x = a \times y \wedge y < b\} \texttt{ x := x+a ; y := y+1 } \{x = a \times y \wedge y \leq b\}}\ \mathsf{seq}$$

   Pour trouver S, on calcule la pré-condition nécessaire pour appliquer la règle de l'affectation à droite.

   $$\begin{aligned} x = a \times y \wedge y \leq b[y \mapsto y + 1] &\Leftrightarrow x = a \times y + 1 \wedge y + 1 \leq b \\ &\Leftrightarrow x = a \times y + 1 \wedge y < b \end{aligned}$$

   On a $y + 1 \leq b \Leftrightarrow y < b$ car on est dans les entiers. Pas besoin de règle de conséquence pour cette équivalence triviale.
   On note $S \Leftrightarrow x = a \times y + 1 \wedge y < b$ et on applique la règle de l'affectation à droite.

On vérifie ensuite qu'avec S en post-condition, on peut appliquer la règle de l'affectation à gauche.

$$\begin{aligned} S[x \mapsto x+a] &\Leftrightarrow x = a \times y + 1 \wedge y < b[x \mapsto x+a] \\ &\Leftrightarrow x + a = a \times (y+1) \wedge y < b \\ &\Leftrightarrow x + a = a \times y + a \wedge y < b \\ &\Leftrightarrow x = a \times y \wedge y < b \end{aligned}$$

Donc on applique la règle de l'affectation à gauche et la preuve est terminée.

$$\dfrac{\dfrac{}{\{x = a \times y \wedge y < b\}\ \texttt{x := x+a}\ \{S\}}\ \text{aff} \qquad \dfrac{}{\{S\}\ \texttt{y := y+1}\ \{x = a \times y \wedge y \leq b\}}\ \text{aff}}{\{x = a \times y \wedge y < b\}\ \texttt{x := x+a ; y := y+1}\ \{x = a \times y \wedge y \leq b\}}\ \text{seq}$$

5. (barème : 0,5)
Ce triplet s'obtient à partir de la conclusion de la question précédente en appliquant la règle du `WHILE`. Pour vérifier que les conditions d'application de cette règle sont satisfaites, on calcule $P \wedge \neg cond$, avec P la pré-condition du triplet à prouver dans cette question et *cond* la condition de la boucle.

$$\begin{aligned} P \wedge \neg cond &\Leftrightarrow x = a \times y \wedge y \leq b \wedge \neg y < b \\ &\Leftrightarrow x = a \times y \wedge y = b \end{aligned}$$

On peut donc appliquer la règle du `WHILE` et on obtient en prémisse la conclusion de la question précédente.

$$\dfrac{\{x = a \times y \wedge y < b\}\ \texttt{x := x+a ; y := y+1}\ \{x = a \times y \wedge y \leq b\}}{\{x = a \times y \wedge y \leq b\}\ \texttt{Loop}\ \{x = a \times y \wedge y = b\}}\ \text{while}$$

On remarque que la formule $x = a \times y \wedge y \leq b$ est bien préservé par la boucle.

6. (barème : 1)
Ce triplet s'obtient à partir de la conclusion de la question précédente en appliquant la règle de conséquence. Pour vérifier qu'elle s'applique, on doit montrer deux implications : (1) $P \Rightarrow Inv$ et (2) $Inv \wedge \neg cond \Rightarrow Q$, où P est la pré-condition du triplet à prouver dans cette question et q sa post-condition.
(1) Si $b \geq 0 \wedge x = 0 \wedge y = 0$ alors on a bien $x = a \times 0 = 0$ et $y \geq 0$.
(2) $x = a \times y \wedge y = b \Leftrightarrow x = a \times b$
Donc on applique la règle de conséquence pour introduire l'invariant, et on obtient en prémisse la conclusion de la question précédente.

$$\dfrac{P \Rightarrow Inv \quad \{x = a \times y \wedge y \leq b\}\ \texttt{Loop}\ \{x = a \times y \wedge y = b\} \quad Inv \wedge \neg cond \Rightarrow Q}{\{x = a \times y \wedge y \leq b\}\ \texttt{Loop}\ \{x = a \times y \wedge y = b\}}\ \text{cons}$$

Donc Inv est bien l'invariant de la boucle.

7. (barème : 0,5)
Variant pour prouver la terminaison de la boucle : $b - y$.
En effet, $b - y = b \geq 0$ avant la boucle, et comme $y \leq b$ (invariant), $b - y$ reste positif ou nul à chaque tour de boucle. De plus, il décroît strictement à chaque tour de boucle puisque $b - (y+1) = b - y - 1 < b - y$.

Exercice 3 (barème : 3,5)

1. (barème : 1,5)

   Il n'y a aucune pré-condition pour pouvoir exécuter cette fonction. La post-condition a deux parties : soit la fonction renvoie *true* et il existe un indice du tableau où se trouve v, soit la fonction renvoie $false$ et toutes les valeurs du tableau sont différentes de v.

   Pré-condition : $true$

   Post-condition : $(result = true \wedge (\exists i, 0 \leq i < t.length \wedge t[i] = v))$
   $\vee\ (result = false \wedge (\forall i, 0 \leq i < t.length \Rightarrow t[i] \neq v))$

   On peut écrire cette post-condition en une seule ligne avec une équivalence :

   Post-condition : $result \Leftrightarrow (\exists i, 0 \leq i < t.length \wedge t[i] = v)$

2. (barème : 1,5)

   La partie la plus importante de l'invariant explique que si le parcours du tableau continue, c'est que v n'a toujours pas été trouvé. Donc si on continue le parcours, c'est que toutes les valeurs du tableau aux indices précédant i sont différentes de v (i exclus puisqu'on n'a pas encore regardé $t[i]$).

   L'autre partie de l'invariant assure qu'on va bien parcourir tout le tableau jusqu'au bout puisque i prend toutes les valeurs de 0 à $t.length$ inclus ($i = t.length$ en sortant de la boucle).

   Invariant : $(\forall j, 0 \leq j < i \Rightarrow t[j] \neq v) \wedge 0 \leq i \leq t.length$

3. (barème : 0,5)

   Variant pour prouver la terminaison de la boucle : $t.length - i$.

   En effet, comme le dit l'invariant, i est toujours inférieur à $t.length$ donc $t.length - i \geq 0$. De plus, il décroît strictement à chaque tour de boucle puisque i est incrémenté à chaque tour de boucle.