université de BORDEAUX

Année Universitaire 2021/2022
Seconde Session de Printemps 2022

Parcours : Master 1 Informatique
Code UE : 4TIN801U
Épreuve : Conception Formelle
Date : Jeudi 30 juin 2022 **Heure :** 11 heures
Durée : 1 heure 30
Documents : non autorisés
Épreuve de M. Vincent Penelle

Collège Sciences et Technologies

Indiquez votre code d'anonymat :

Avertissement

- Les deux exercices sont indépendants, et dans chacun d'entre eux, certaines questions dépendent des précédentes.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).
- Le barème est donné à titre indicatif, et pourra donc être différent d'indiqué. Chaque question d'un exercice comporte un nombre d'étoiles représentant sa longueur/difficulté et donc son poids relatif envisagé dans la note. Elles n'ont pas la même signification dans les deux exercice (le premier étant plus court et facile que le second).
- Les questions bonus sont facultatives (mais peuvent rapporter des points).

Question	Points	Score
Rappels et compréhension	6	
maxConsecutive	14	
Total:	20	

Rappel:

Un triplet de Hoare est un tuple de la forme $\{\varphi\}$ `P` $\{\psi\}$ où `P` est un *programme* (dans la suite, on considérera toujours qu'il s'agit d'une suite d'instructions en C, ou d'une fonction complète), et ψ et φ sont des formule de la logique du premier ordre dont les *variables libres* (i.e., non capturées par un quantificateur) représentent des variables du programme (qui doivent exister à la position où la formule est évaluée). φ est la *pré-condition* du triplet et ψ est sa *post-condition*. Sur Frama-C, la pré-condition d'une fonction est la conjonction des clauses `requires`, et la post-condition est la conjonction des clauses `ensures`, ainsi que des clauses `assigns`. Une clause `assert` indique une post-condition du *début* de la fonction jusqu'à l'instruction après laquelle apparaître la clause `assert`.

La *vérification* d'un programme consiste à démontrer la validité d'un triplet de Hoare grâce au calcul de plus faible précondition en calculant WP(P,ψ), puis en démontrant que $\varphi \Rightarrow WP(\texttt{P}, \psi)$ est une formule valide (i.e., vraie quelque soit la valeur des variables libres).

On rappelle les règles du calcul de weakest precondition, dans laquelle `t` représente une expression arithmétique, $\texttt{e}_1$ et $\texttt{e}_2$ des expressions en C, et φ une formule de la logique du premier ordre ayant pour variables libres les variables du programme :

- WP(`skip`, φ) = φ.
- WP(`int x`, φ) = φ.

— WP(`x=t`, φ) = φ[x ← t].
— WP(`return t;`, φ) = φ[\result ← t].
— WP(`e1;e2`, φ) = WP(`e1`, WP(`e2`, φ)).
— WP(`if(t) e1; else e2;`,φ) = t ⇒ WP(e_1, φ) ∧ ¬ t ⇒ WP(e_2, φ).

On rappelle que $\varphi[x \leftarrow t]$ est la formule φ dans laquelle on a remplacé toutes les occurrences de x par le terme t. Par exemple $(x == 2 \vee 3 * x \geq y - x)[x \leftarrow y + x])$ est la formule $(y + x == 2 \vee 3 * x + 3 * y \geq -x)$.

On rappelle également que si la formule φ contient un \old(x), alors dans WP(P,φ), on remplace \old(x) par x. De même, \at(x,i) est remplacé par x dans WP(i-j,φ).

Pour les boucles, il est impossible de calculer la plus faible précondition directement. Il faut fournir (manuellement) un *invariant de boucle* et démontrer qu'il permet, c'est-à-dire une formule préservée par l'effet de la boucle (i.e., si la formule est vraie à l'entrée d'un tour de boucle, elle sera vraie après). Pour démontrer le contrat, il faut de plus montrer que, avec la condition de sortie de boucle, cet invariant permet de déduire la post-condition que l'on cherche à prouver, et enfin démontrer qu'il est vérifié au tout début de la boucle (il est établi).

Plus formellement, on peut voir cela comme la règle suivante :
$WP($`while(test)@I@ B;`$, \psi)) = I$, si les deux propriétés suivantes sont vraies :
— `test` $\wedge\, I \Rightarrow WP(B, I)$ est valide (invariant)
— ¬`test` $\wedge\, I \Rightarrow \psi$ est valide (implique la post-condition).

On peut découper l'invariant en plusieurs invariants qu'on prouvera indépendamment dont on prendra la conjonction (en utilisant que $\varphi \Rightarrow (\psi_1 \wedge \psi_2)$ et $(\varphi \Rightarrow \psi_1) \wedge (\varphi \Rightarrow \psi_2)$ sont équivalentes).

En Frama-C, l'invariant est la conjonction des clauses `loop invariant` (plus la clause `loop assigns`, qu'il vous sera demandé d'expliciter dans les questions).

Un variant de boucle est un terme arithmétique V à valeur entière tel que :
— V est positif au début de chaque tour de boucle (il faut donc démontrer que I ∧ t ⇒ V ≥ 0).
— Un tour de boucle diminue strictement la valeur de V (il faut donc démontrer que `test`∧ I ⇒ WP(`B`,\at(V,i) - V > 0) où i est la première ligne du corps de la boucle B).

En Frama-C, un variant de boucle est indiqué par la clause `loop variant`.

Enfin, toujours en Frama-C, lorsqu'on déclare un *prédicat* (c'est-à-dire une formule logique qui sera utilisée ailleurs – dans différents contextes) il est possible de lui spécifier des labels comme arguments qui permettront d'évaluer ses arguments qui représentent des variables du programmes à différents labels. Par exemple, il sera possible de définir un prédicat `IsEqual{L,M}(integer x) = \at(x,L) == \at(x,M)`, et de l'utiliser dans le programme, par exemple comme `assert IsEqual{Pre,Here}(y);` si on veut démontrer que la variable `y` a la même valeur à la position courante qu'avant l'appel de la fonction. C'est notamment indispensable lorsqu'on veut utiliser les valeur pointées par un même pointeur (ou tableau) à différents labels.

Exercice 1 : Rappels et compréhension (6 points)

(a) (*) Décrivez en français la signification de la validité d'un triplet de Hoare de la forme $\{\varphi\}$ `P` $\{\psi\}$ (i.e., quand le triplet est valide, que peut-on dire sur l'effet de l'application du programme P dans un état de mémoire M).

(b) (***) Qu'est-ce qu'un comportement indéterminé pour une instruction C ? Donnez deux exemples d'instructions pouvant avoir un comportement indéterminé, ainsi que l'état de mémoire le provoquant : un provoquant une erreur à l'exécution si compilé avec gcc, et un ne provoquant pas d'erreur à l'exécution, et en donnant le résultat obtenu après le comportement indéterminé si compilé avec gcc. Est-ce qu'avec un autre compilateur respectant la norme C vous auriez nécessairement le même résultat dans ces exemples ?

(c) (**) Que doit-on faire pour vérifier qu'un programme ne provoque pas de comportements indéterminés avec le calcul de weakest precondition ? Quel module de Frama-C est responsable de cela ?

(d) (**) On considère une fonction `int toto(int n, int *t)`. Exprimez les propriétés suivantes sur cette fonction en logique du premier ordre.

1. Le résultat est plus grand que tous les $t[i]$ pour i entre 0 et $n-1$ (inclus).
2. Le résultat est une valeur du tableau t (entre 0 et $n-1$).
3. Tous les $t[i]$ (pour i entre 0 et $n-1$) ne sont pas égaux (i.e., au moins deux d'entre eux sont différents).
4. Tous les $t[i]$ qui sont pairs sont strictement plus petits que leur voisin de droite dans le tableau (s'il existe).

(e) (**) On considère une fonction `int titi(int* a, int * b)`. On suppose qu'un certain triplet de Hoare $\{\varphi\}$ `P` $\{\psi\}$ a été démontré valide pour cette fonction.

On considère un pointeur `int* c`, et on suppose qu'on est dans un état de mémoire M qui satisfait $\varphi[a \leftarrow c][b \leftarrow c]$.

Que peut-on dire sur le résultat de l'appel `titi(c,c)` ? Justifiez.

Donnez un exemple de fonction et de triplet pour lesquels on observe ce phénomène.

Exercice 2 : maxConsecutive (14 points)

```
1   int maxConsecutive(int *t, int size, int val)
2   {
3       int max = 0;
4       int pos = 0;
5       int cur = 0;
6       while (pos < size)
7       {
8           if (t[pos] == val)
9               cur++;
10          else
11              cur = 0;
12          if (max < cur)
13              max = cur;
14          pos++;
15      }
16      return max;
17  }
```

Informellement, cette fonction va trouver la taille de la plus longue plage de valeurs toutes égales à `val` dans le tableau `t`.

Plus formellement, on considère les post-conditions suivantes :

`ConsecutiveRange` := $\exists p; 0 \leq p \leq size - \texttt{\result} \wedge \forall i; p \leq i < p + \texttt{\result} \Rightarrow t[i] = val;$

`Result` := $0 \leq \texttt{\result} \leq size;$

Le but de cet exercice est de trouver une pré-condition permettant au programme de valider la post-condition précédente, et d'en démontrer la correction.

Dans vos calculs de weakest precondition, vous pourrez représenter des instructions par leur numéros de lignes (e.g., écrire WP(14,φ) à la place de WP(pos++,φ)), et vous pourrez également utiliser les numéros de lignes pour représenter des sections du programmes, par exemple WP(8-14,φ) désignera le sous-programme commençant avant l'instruction de la ligne 8 et s'arrêtant après l'instruction de la ligne 14.

Comme nous sommes dans une preuve sur papier, nous faisons les mêmes hypothèses que dans le cours, à savoir que nous considérons que les variables sont de vrais entiers de $\mathbb{Z}$ et pas des entiers machine. On considèrera également qu'aucun comportement indéterminé ne peut survenir.

(a) (*) Commencez par calculer WP(16,`Result`) et WP(16,`ConsecutiveRange`) (comme le résultat est la simple application d'une règle, il est inutile de détailler ici).

(b) (*) Nous allons maintenant commencer à définir des invariants, et démontrer qu'ils impliquent bien notre post condition.

Donnez la clause `loop assigns` appropriée à la boucle.

(c) (**)

Déterminez trois invariants :

I1 qui est l'invariant lié à la condition de fin de boucle

IRes qui est un invariant permettant d'impliquer WP(16,`Result`)

ICR qui est un invariant permettant d'impliquer WP(16,`ConsecutiveRange`)

Justifiez que $\neg(pos < size) \wedge I1 \wedge IRes \Rightarrow Result$ et que $\neg(pos < size) \wedge I1 \wedge ICR \Rightarrow ConsecutiveRange$ (la justification est commune aux deux et est très syntaxique).

Pour le moment, on admettra que les invariants que vous venez de donner sont suffisant pour démontrer la correction de cette fonction (ce n'est pas tout à fait le cas, mais on le verra plus tard en démontrant ces invariants).

(d) (**) Calculer WP(3-5,φ) pour une formule φ quelconque.
Déduisez-en WP(3-5,I) pour chaque invariant et déduisez-en le triplet de Hoare que nous venons de démontrer.

(e) (*) Trouvez une fonction `fakeMaxConsecutive` très simple (une ligne) pour laquelle le même triplet sera valide. Prouvez-le en calculant WP(fakeMaxConsecutive,Result $\wedge$ Consecutive-Range). On rappelle qu'un triplet $\{\varphi\}P\{\psi\}$ est valide si $WP(P, \psi) \Rightarrow \varphi$.

(f) (***) Quelle post-condition manque-t'il pour séparer ces deux fonctions ? Exprimez-la en français (moitié des points), puis en logique. On appellera ψ cette propriété mystère.

(g) (***) Calculez WP(16,ψ) et déterminez un invariant I_ψ qui permet d'impliquer cette formule quand $\neg(pos < size)$ et les autres invariants sont vrais (i.e., en fin de boucle). Justifiez que cette implication est vraie.

On va maintenant chercher à démontrer que les formules déterminées précédemment sont des invariants, pour démontrer qu'on avait bien le droit de faire le raisonnement que l'on a fait (c'est le cas, mais modulo quelques caveats).

Si on calcule WP(8-14,φ) pour une formule quelconque φ, on obtient :

$$\begin{aligned}\text{WP(8-14,}\varphi\text{)} := \quad & ((t[P] = V \wedge M < C + 1) \Rightarrow \varphi[P \leftarrow P + 1][M \leftarrow C][C \leftarrow C + 1]) && \text{(A)}\\ & \wedge((t[P] = V \wedge M \geq C + 1) \Rightarrow \varphi[P \leftarrow P + 1][C \leftarrow C + 1]) && \text{(B)}\\ & \wedge((t[P] \neq V \wedge M < 0) \Rightarrow \varphi[P \leftarrow P + 1][M \leftarrow C][C \leftarrow 0]) && \text{(C)}\\ & \wedge((t[P] \neq V \wedge M \geq 0) \Rightarrow \varphi[P \leftarrow P + 1][C \leftarrow 0]) && \text{(D)}\end{aligned}$$

où on a remplacé val par V, max par M, cur par C et pos par P (pour faire plus court, vous pouvez garder cette convention, avec S pour size aussi). Et on a nommé chacune de ces implications (à droite).

(h) (*) Justifiez que l'une de ces implications sera toujours impliquée par $IRes$.

Dans la suite du sujet, vous traiterez chacune de ces trois implications indépendamment dans vos calculs et justifications. Je vous conseille fortement de les écrire de la manière suivante : «pour (A), $IRes$[...] donne la formule [...]. Elle est bien impliquée par IRes et a et b (et ..) car [...]».

Ce que vous avez à montrer en faisant ça, c'est une formule du genre $P < S \wedge I \Rightarrow (t[P] = V \wedge M < C + 1) \Rightarrow \varphi[...])$ (pour (A)). Mais, comme $\psi_1 \Rightarrow (\psi_2 \Rightarrow \psi_3)$ est équivalente à $(\psi_1 \wedge \psi_2) \Rightarrow \psi_3$, vous avez sacrément intérêt à considérer que tout ce qui est à gauche de l'implication la plus à droite sont des hypothèses avec lesquels vous allez démontrer φ[...] (par exemple, grâce à $IRes$ et $M < C + 1$, je déduis que [...]). Pour ça vous avez deux cas types : soit les hypothèses contiennent une contradiction (donc la formule est vraie), soit les hypothèses permettent bien de démontrer φ[...].

Vous pourrez écrire «$IRes$[...]» sans remettre les substitutions (elles sont écrites dans cette question, inutile de le recopier partout). Quand deux substitutions donnent le même résultat (ça arrive souvent), n'écrivez la formule qu'une fois. Le message principal est que toutes ces questions sont similaires, économisez-vous du temps d'écriture. On n'attendra pas de preuve entièrement formelle, mais une justification en français convaincante. Vous pouvez cela dit, découper vos formule pour exhiber des sous-formules triviales (particulièrement pour ICR). N'oubliez pas de simplifier φ[...] le plus possible. Les preuves sont à peu près toutes assez syntaxiques.

(i) (**) On va d'abord montrer que la fonction termine. Déterminez un variant Var pour la

boucle, et démontrez que $pos < size \Rightarrow Var \geq 0$ et que $WP(8-13, Var - \text{at}(Var, 8) < 0)$ est vraie (pas besoin d'implications ici). Pour cette dernière question, vous pouvez comme aux questions précédente vous contenter de donner $(Var - \text{at}(Var, 8) < 0)[...]$ pour les trois implications de la question h et justifier qu'il est vrai (cela devrait donner la même formule pour les trois implications).

(j) (*) Donnez $I1[...]$ dans les trois implications précédentes.
Justifier que $pos < size \wedge I1 \wedge IRes \Rightarrow WP(8-14, I1)$.

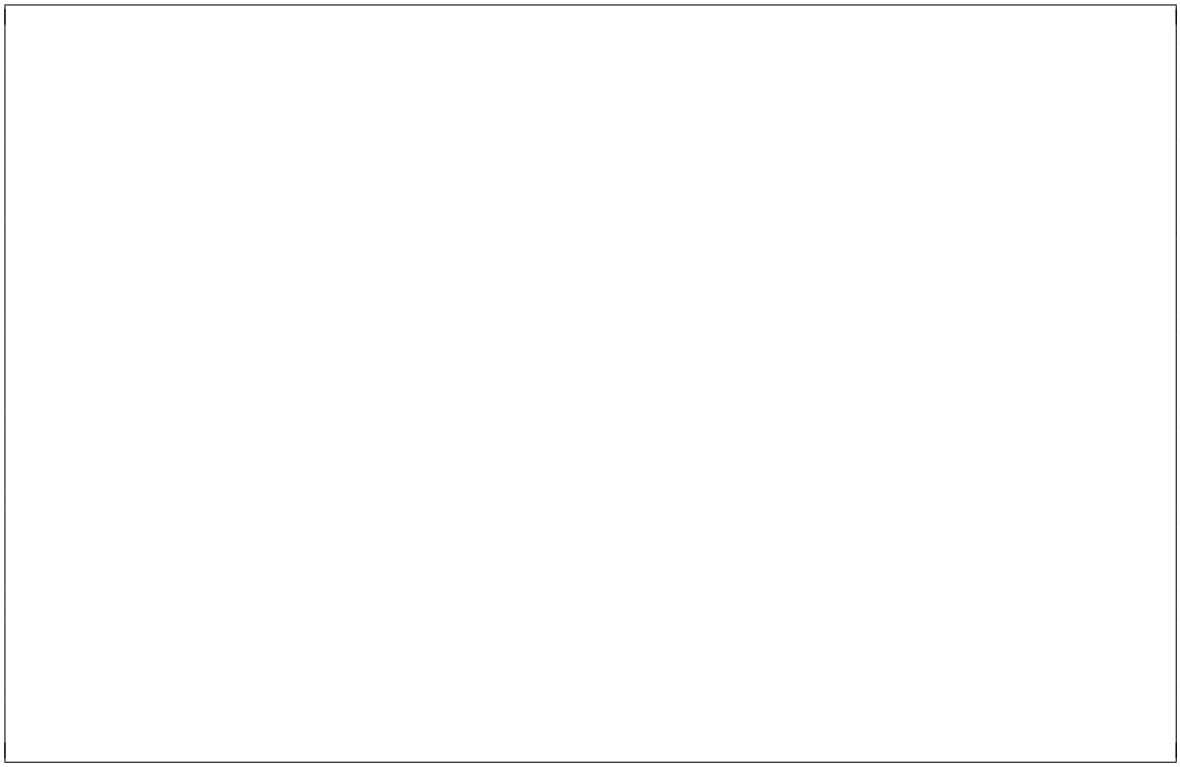
(k) (*) Donnez $IRes[...]$ dans les trois implications de la question h.
Justifier que $IRes$ implique deux des implications obtenues.

(l) (**) Déterminez une formule $IRes2$ qui permet d'impliquer l'implication précédente (justifiez que c'est le cas).

On ne demande pas de démontrer que IRes2 est un invariant (mais il doit en être un). (Bonus si vous le démontrez – seulement si vous avez du temps)

(m) (**) Donnez $ICR[...]$ dans les trois implications de la question h.
Justifiez que deux des implications obtenues sont impliquées par ICR.

(n) (****) Déterminez une formule $ICR2$ qui permet d'impliquer l'implication précédente (justifiez que c'est le cas).
On ne demande pas de démontrer qu'ICR2 est un invariant (mais il doit en être un). (Bonus si vous le démontrez – seulement si vous avez du temps)



On admettra que I_ψ est bien un invariant. On aurait pu le donner en bonus, mais ça me semble faire assez de questions là.