Faculté des sciences
Aix Marseille Université

Année 2025-26

Domaine Sciences et Technologies
Master informatique
Fiabilité logicielle : TD 4
Codes UE : SINB34A
Preuves de programmes

1 Triplets de Hoare

Question 1.1 : Démontrer la validité du triplet de Hoare suivant :
$\{true\}$ if (c != 0) then {i := 0; x := i} else x := c $\{x = 0\}$

Question 1.2 : Démontrer la validité du triplet de Hoare suivant :
$\{true\}$N := 10; s := 0; i := 1; while i <= N do s := s + i ; i := i + 1 done$\{s = 55\}$

Pour mémoire, la somme des n premiers entiers vaut $n(n+1)/2$.

2 Preuve de programmes

On considère le programme suivant.

```
// calcul de factorielle de i, resultat dans n
n := 1;
j := i;
while (j > 1) {
   n := n * j;
   j := j - 1;
}
```

Question 2.1 : Montrer que `n * j! = i!` est un invariant de la boucle mais qu'il ne permet pas de vérifier que le calcul est correct.

Question 2.2 : Proposer un renforcement de cet invariant (sous la forme d'un autre invariant) qui permet de conclure.

Question 2.3 : Donner un variant prouvant la terminaison.

3 Recherche dichotomique

Voici le code de la recherche dichotomique d'un élément `target` dans le tableau `t` trié. La fonction retourne l'indice d'une case où se trouve `target` s'il est présent dans le tableau est -1 sinon.

```
int dichotomicSearch (int * t, int size, int target){
   int low := 0;
   int high := size -1;
   while ( low < high ){
      int mid := ( low + high )/2;
      if ( tab [ mid ] < target ){
         low := mid + 1;
      }
      else {
         high := mid;
      }
   }
   if ( tab [ low ] == target ){
      res := low;
   }
   else {
      res := -1;
   }
   return res;
}
```

Question 3.1 : Donner une définition pour le prédicat `IsSorted(t)` spécifiant que le tableau `t` est trié.

Question 3.2 : Donner une définition pour le prédicat `IsMember(t, e)` spécifiant que l'élément `e` est présent dans le le tableau `t`.

Question 3.3 : Exprimer sous la forme d'une formule la spécification de la fonction.

Question 3.4 : Montrer que :
$0 \leq low \leq high < size \wedge IsSorted(tab) \wedge IsMember(tab, target) \Rightarrow tab[low] \leq target \leq tab[high]$
est un invariant de la boucle de la fonction.

4 Annexe : règles de calcul de Weakest Precondition (WP)

- **Règle composition** : $WP((i_1; i_2), Q) = WP(i_1, WP(i_2, Q))$
- **Règle affectation** : $WP(x := e, Q) = Q[x \leftarrow e]$ si e est évaluée sans erreur
- **Règle conditionnelle** : $WP(\text{if e then } i_1 \text{ else } i_2, Q) = ((e \Rightarrow WP(i_1, Q) \wedge \neg e \Rightarrow WP(i_2, Q))$
- **Règle boucle** : $WP(\text{while e do i done}, Q) \Leftarrow I$, où I est un invariant de la boucle avec $I \Rightarrow WP(i, I)$ et $I \wedge \neg e \Rightarrow Q$.