Licence Info
Algorithmique Avancée
TD 9 : décembre 2006
Invariants de boucles

1. Pour la boucle donnée, répondre par oui ou non aux questions suivantes pour chacune des expressions données:

   (a) Est-elle une expression booléenne forcément vraie (si la précondition était vraie) à chaque commencement du calcul de la condition de la boucle?

   (b) Est-elle une conséquence de la précondition?

   (c) Est-elle forcément vraie à la fin d'une itération de la boucle si elle l'était au début (et que la condition de la boucle était vraie)?

   (d) Garantit-elle (étant donné que la condition de la boucle est vraie) que l'itération peut être exécutée sans erreur?

   (e) Suffit-elle (avec la négation de la condition de la boucle) pour entraîner la postcondition?

   /* calcul de la partie la plus longue strictement croissante d'une suite donnée dans un tableau [1..n] ($n > 1$) */

```
/* précondition: l=1 b=1 i=2 */
tant que i<=n
   si T[i]>T[i-1] alors
      l:=l+1
      si l>b alors
         b:=l
      finsi
   sinon l=1
   finsi
   i:=i+1
ftq
/* postcondition b est la longueur de la plus longue section croissante de
T[1..n] */
```

   (a) $l - b$

   (b) $i <= n$

   (c) $i >= 2$

   (d) b est la longueur de la plus longue partie croissante de $T[1..i]$

   (e) l est la longueur de la plus longue partie croissante de $T[1..i-1]$ et qui finit $T[i-1]$

   (f) $i >= 1$ et la condition précédente

Donner un invariant pour laquelle toutes les réponses auraient été OUI.

2. On considère l'algorithme "TriLent" qui trie (en ordre croissant) un tableau T, indicé de 1 à n.

```
i:=1;
tant que i<n faire
   si T[i+1] >= T[i]
      alors i:=i+1;
      sinon
         t:=T[i];T[i]:=T[i+1];T[i+1]:=t;
         i:=1;
   finsi
fintantque;
```

En utilisant la règle de déduction pour une affectation `x:=E;`, à savoir que $A(x)$ est vrai après l'affectation si $A(E)$ était vrai avant, démontrez formellement que l'assertion "la partie du tableau de `T[1]` à `T[i]` est triée en ordre croissant" est vraie après:

   (a) après `i:=i+1;` si elle était vraie avant et aussi `T[i+1] >= T[i]`,
   (b) après `t:=T[i];T[i]:=T[i+1];T[i+1]:=t; i:=1;` si elle était vraie avant, sans autres conditions

3. Démontrez que "`T` est une permutation de `T0`" est un invariant de la boucle (T0 étant l'ensemble initial des valeurs de T).

4. fusion de deux listes chaînées
Donnez un invariant:
/* précondition : chaque liste liste1 et liste2 est triée en ordre croissant */

```
p:=liste1 q:=liste2
fu:=nul
tant que p!=nul ou q!=nul
   si fu=nul alors
      allouer(fu);v:=fu
   sinon
      allouer(nouveau)
      v^.prochain := nouveau
      v:= nouveau
   finsi
   si q=nul ou (p!=nul et p^.valeur<q^.valeur) alors
      v^.valeur:=p^.valeur; p:=p^.prochain
   sinon
      v^.valeur:=q^.valeur; q:=q^.prochain
   finsi
```

```
ftq
si fu!=nul alors v^.prochain:=nul
/* postcondition : fu pointe vers une nouvelle liste formée des éléments
de liste1 et liste2 en ordre croissant */
```

5. Donner un invariant de la boucle suivante qui calcule le *pgcd* de x et y et, en même temps les valeurs m, n telles que ce *pgcd* est égal à $mx + ny$.

```
debut
   x0:=x;
   y0:=y;
   m:=1;
   n:=0;
   k:=0;
   l:=1;
   tant que x!=y faire
      si x>y alors
         x:=x-y;
         m:=m-k;
         n:=n-l
      sinon
         si y>x alors
            y:=y-x;
            k:=k-m;
            l:=l-n
         finsi
      finsi
   fintantque;
   retourne x
fin
```

Modifier la boucle de façon qu'elle calcule les mêmes résultats par divisions au lieu de soustractions.