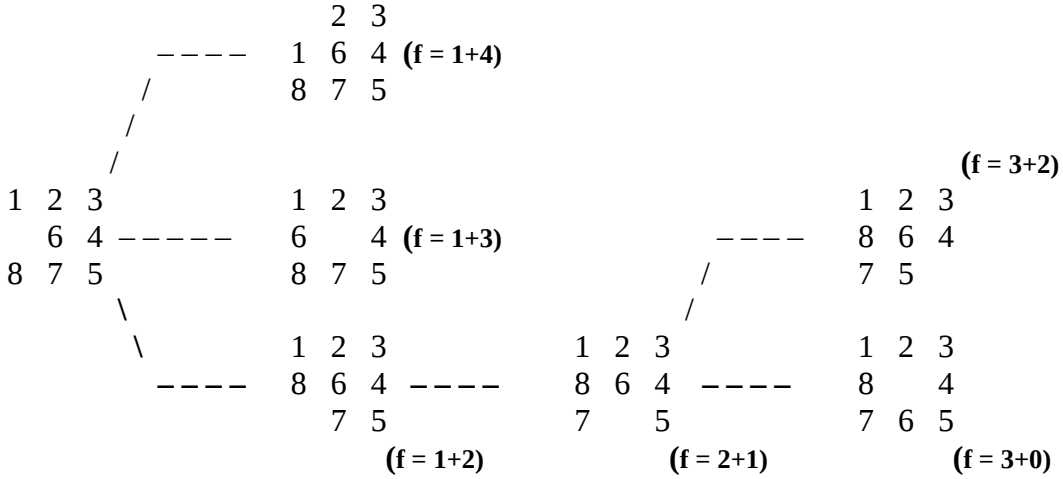
Corrigé Examen Semestriel – TPGO – 2CS – ESI 2014/2015

Durée 2h – Doc. Interdits – Barèmes (3+4+4+6+3) – Justifiez toutes vos réponses

1*- Donnez l'arbre représentant l'espace de recherche exploré par A* pour le problème du taquin. L'état initial et l'état solution sont comme suit :*

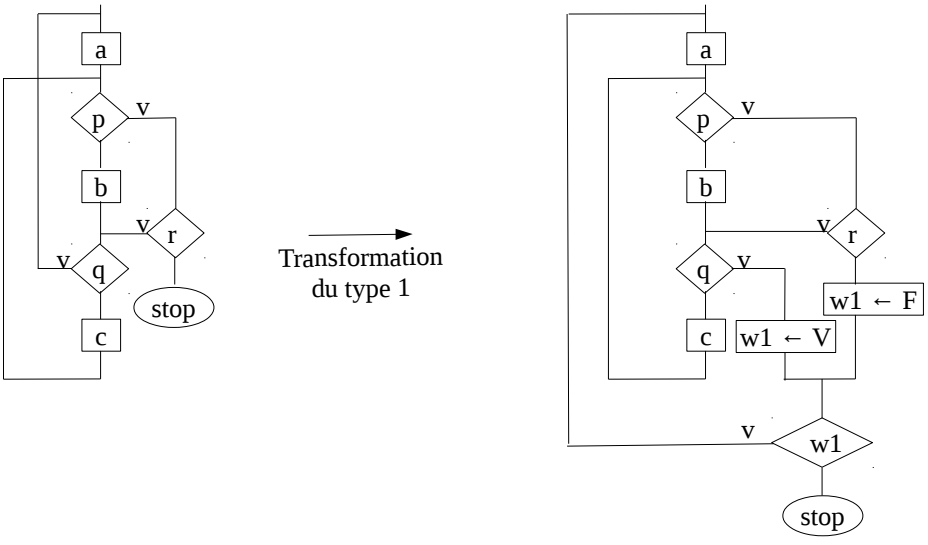
1	*2*	*3*			*1*	*2*	*3*
	6	*4*	→		*8*		*4*
8	*7*	*5*			*7*	*6*	*5*

La fonction d'estimation représente le nombre de pièces mal-placées.

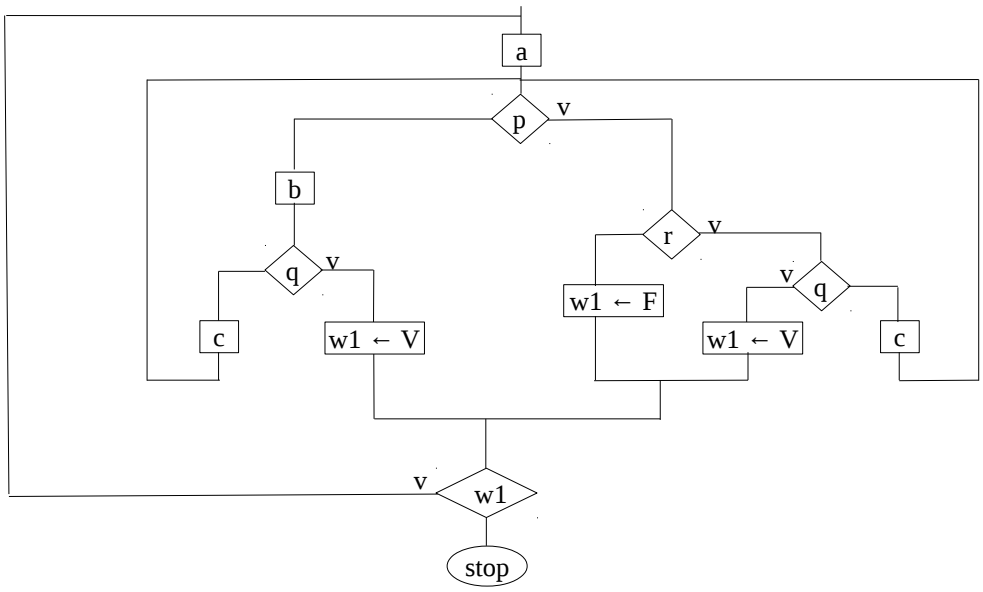


2*- En utilisant le théorème de Bohm et Jacopini, donnez un programme structuré (D-algorithme), fonctionnellement équivalent au programme ci-dessous :*

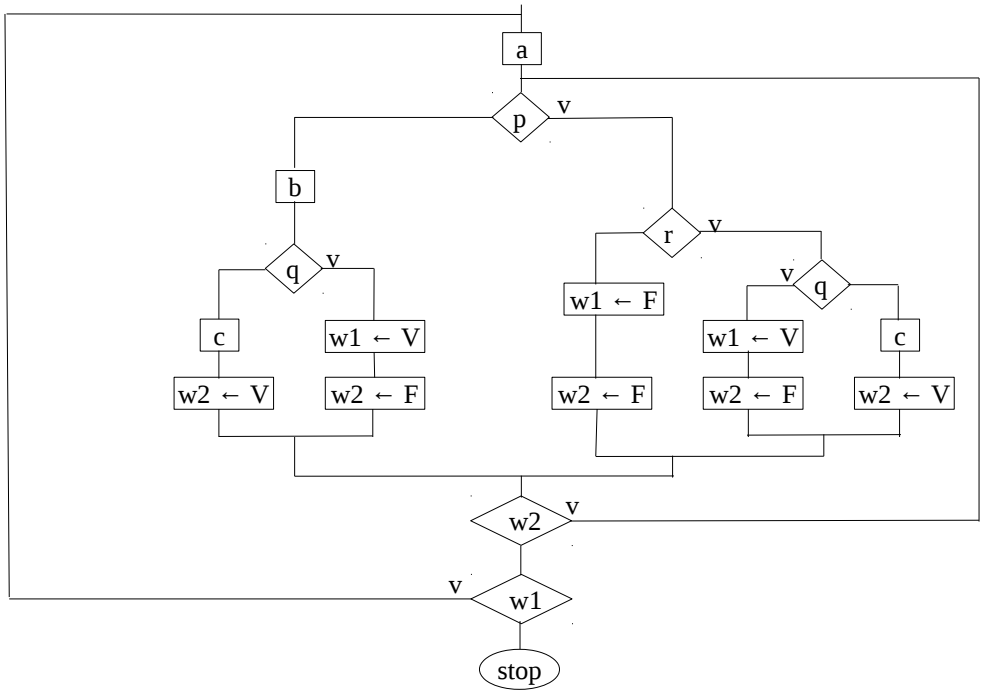
et1 : a ;
et2 : SI p Aller à et4 ;
b ;
et3 : SI q Aller à et1 ;
c ;
Aller à et2 ;
et4 : SI r Aller à et3 ;
stop



Le sous-organigramme commençant par le test p et se terminant par l'entrée du test w1, est de de type 3. En dupliquant les parties communes, on le transforme en type 2 :



Une transformé en type 2, on applique les règles du type 2 pour le rendre structuré :



3*- Soit T[1..n] un tableau de n entiers (variable global) et soit P le corps de la procédure récursive suivante :*

```
min( i, j : entier ; var r:entier )          // i et j des entrées et r une sortie
SI ( i = j )      r ← T[i]
SINON
                  min( i+1, j, r ) ;
                   SI ( T[i] < r )   r ←  T[i]   FSI
FSI
```

Montrez à l'aide du système formel de Hoare, que cette procédure retourne dans r le plus petit élément du sous-tableau T[i .. j] (c-a-d le sous-tableau formé par les éléments entre les indices i et j, avec i<= j). Définir préalablement les conditions d'entrée et de sortie (E et S).

Pour exprimé que la procédure min(i,j,r) calcule, dans r, la valeur minimale du tableau T entre les indice i et j, on peut proposer les conditions d'entrée et de sortie suivantes :

E : [$1 \le i \le j \le n$] et S : [$\forall$ k, ($(i \le k \le j) \Rightarrow r \le T[k]$)]

Posons P le corps de la procédure min(i,j,r) et P1 le bloc : [min(i+1, j, r) ; SI (T[i] < r) r ← T[i] FSI].

L'énoncé E{p}S sera vrai, d'après la règle CND2 si :
(a) E & i=j { r ← T[i] } S et (b) E & i≠j { P1 } S

L'énoncé (a) E & i=j { r ← T[i] } S est vrai par la règle IMP1 car ses 2 prémisses sont vraies :
(a1) (E & i=j) $\Rightarrow$ $\forall$ k, ($(i \le k \le j) \Rightarrow T[i] \le T[k]$) et
(a2) $\forall$ k, ($(i \le k \le j) \Rightarrow T[i] \le T[k]$) { r ← T[i] } S

(a2) est vrai car c'est un axiome de l'affectation (AFF)
(a1) est vrai car si i=j, alors si k=i=j on aura donc T[i] = T[k] et donc $T[i] \le T[k]$

L'énoncé (b) E & i≠j { P1 } S est vrai par SEQ, car :
(b1) E & i≠j { min(i+1, j, r) } F et
(b2) F { SI (T[i] < r) r ← T[i] FSI } S
le sont aussi comme démontrés ci-dessous :

(b1) est vrai par IMP1, car d'une part :
l'implication $(1 \le i \le j \le n) \& i \ne j \Rightarrow (1 \le i+1 \le j \le n)$ est vraie,
et d'autre part :
l'énoncé $(1 \le i+1 \le j \le n)$ { min(i+1, j, r) } F est aussi vrai <u>par hypothèse de validité des appels internes</u> à une procédure récursive. Il suffit pour cela de choir pour F la condition : $\forall$ k, ($(i+1 \le k \le j) \Rightarrow r \le T[k]$).

(b2) F { SI (T[i] < r) r ← T[i] FSI } S est alors vrai par CND1, car ses 2 prémisses sont vraies aussi :
(b21) F & (T[i] < r) { r ← T[i] } S et (b22) F & ($T[i] \ge r$) $\Rightarrow$ S

(b22) $\forall$ k, ($(i+1 \le k \le j) \Rightarrow r \le T[k]$) & ($T[i] \ge r$) $\Rightarrow$ $\forall$ k, ($(i \le k \le j) \Rightarrow r \le T[k]$) est vrai car si $r \le T[k]$ pour tout k entre i+1 et j et si de plus $r \le T[i]$, donc on aura bien $r \le T[k]$ pour tout k entre i et j.
(b21) est vrai par IMP1 car, d'une part :
l'implication : $\forall$ k, ($(i+1 \le k \le j) \Rightarrow r \le T[k]$) & (T[i] < r) $\Rightarrow$ $\forall$ k, ($(i \le k \le j) \Rightarrow T[i] \le T[k]$) est vraie, puisque T[i] < r et $r \le T[k]$ pour tout k entre i+1 et j, a pour conséquence que $T[i] \le T[k]$ pour tout k entre i et j. D'autre part :
$\forall$ k, ($(i \le k \le j) \Rightarrow T[i] \le T[k]$) { r ← T[i] } $\forall$ k, ($(i \le k \le j) \Rightarrow r \le T[k]$) est un axiome AFF.

Donc l'énoncé initial E{P}S est vrai.

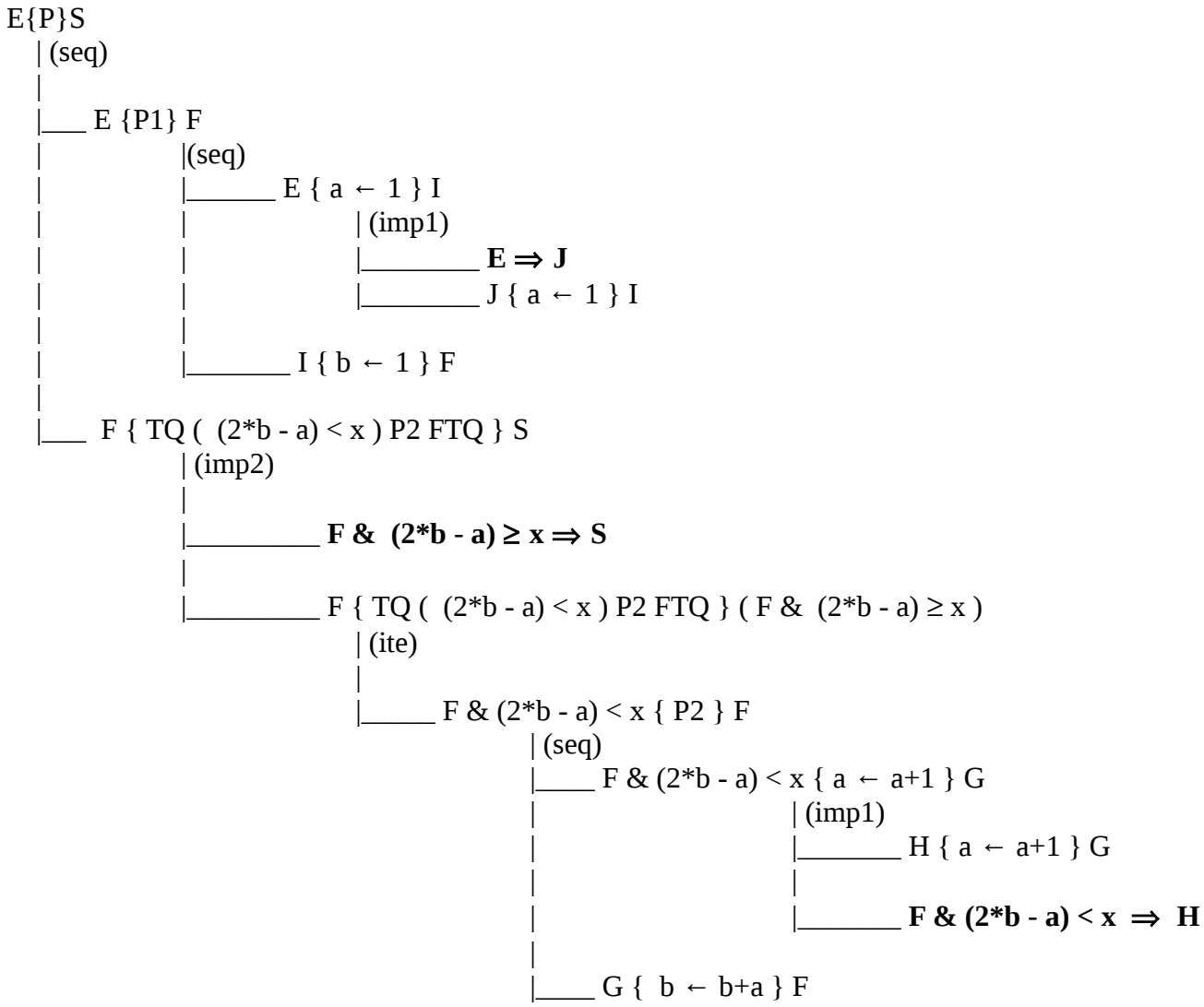
***4**- Soit P le programme suivant :*

a ← 1 ; b ← 1 ;
*TQ ((2*b - a) < x)*
a ← a+1 ; b ← b+a
FTQ

a) Donnez l'arbre de preuve pour la démonstration de l'énoncé : E { P } S, avec E et S des prédicats quelconques. Mettre en évidence l'invariant de boucle.
b) Donnez toutes les implications de l'arbre de preuve qui restent à démontrer.
c) Trouvez le bon invariant de boucle et complétez la preuve dans le cas où :
E vaut ($x \ge 1$) et S vaut ($(a-1)^2 < x$ & $a^2 \ge x$)

Posons P1 = [a ← 1 ; b ← 1 ;], P2 = [a ← a+1 ; b ← b+a]

a) L'arbre de preuve :



L'invariant de boucle est le prédicat F.
Toutes les feuilles de l'arbre de preuve sont soit des axiomes AFF (donc déjà vrais) soit des implications (en gras) qui restent à démontrer.

b) Les implications qui restent à démontrer sont :

i. $E \Rightarrow J$, avec J = [1/a] [1/b] F
ii. $F \;\&\; (2*b - a) \geq x \Rightarrow S$,
iii. $F \;\&\; (2*b - a) < x \Rightarrow H$, avec H = [a+1/a] [b+a/b] F

c) En prenant E : $(x \geq 1)$ et S : $((a-1)^2 < x \;\&\; a^2 \geq x)$, il faudra choisir un bon invariant F qui vérifie les 3 implications (i. ii. et iii.) qui restent à démontrer.
En analysant l'algorithme, on peur remarquer que la variable b contient au début et à la fin de chaque itération, la somme des a premiers entiers (b = 1+2+... a).
Donc b = a (a+1)/2 et de ce fait, l'expression (2b-a) vaut a^2.
L'implication (ii.) peut alors être réécrite comme suit :
$F \;\&\; a^2 \geq x \Rightarrow ((a-1)^2 < x \;\&\; a^2 \geq x)$
Cela suggère alors que la condition $(a-1)^2 < x$ pourrait faire partie de l'invariant F.
D'un autre coté, comme dans le corps de la boucle, la variable b est modifiée à chaque itération et qu'elle est en relation avec la variable a, il faudrait peut être aussi rajouter dans la proposition de l'invariant, une

relation invariante entre ces deux variables : b = a (a+1)/2.
Proposons alors comme invariant **F : $(a-1)^2 < x$ & b = a (a+1)/2** et vérifions les 3 implications de l'arbre de preuve :
- L'implication (i) : E ⇒ J, avec J = [1/a] [1/b] F
c-a-d : $(x \geq 1) \Rightarrow (1-1)^2 < x$ & 1 = 1 (1+1)/2
Cette implication est vraie car $(x \geq 1) \Rightarrow 0 < x$ & 1 = 1
- L'implication (ii) : F & (2*b - a) ≥ x ⇒ S
c-a-d : $(a-1)^2 < x$ & b = a (a+1)/2 & $a^2 \geq x \Rightarrow ((a-1)^2 < x$ & $a^2 \geq x)$
Cette implication est vraie, car toutes les conditions du conséquent ($(a-1)^2 < x$, $a^2 \geq x$) se trouvent déjà dans l'antécédent de l'implication (elle de la forme A & B ⇒ A).
- L'implication (iii) : F & (2*b - a) < x ⇒ H, avec H = [a+1/a] [b+a/b] F
c-a-d : $(a-1)^2 < x$ & b = a (a+1)/2 & $a^2 < x \Rightarrow$ $a^2 < x$ & b+a+1 = (a+1) (a+2)/2
Cette implication est vraie, car l'expression du conséquent b+a+1 = (a+1) (a+2)/2, après simplification, devient : b = a (a+1)/2 (qui se trouve déjà dans l'antécédent), de même que la 1ère condition du conséquent ($a^2 < x$) qui se trouve aussi dans l'antécédent.

Le programme est donc correct, relativement à la précondition E et la postcondition S.

5- En utilisant la théorème du point fixe, établir la fonction exactement calculée par le programme fonctionnel suivant :

f = λn. SI (n > 100) Alors (n-10) Sinon f(f(n+11)) FSI

Il faut donc calculer le plus petit point fixe (f_0) de l'équation :

f = τ(f), avec τ(f) la fonctionnelle : = λn. SI (n > 100) Alors (n-10) Sinon f(f(n+11)) FSI

D'après le théorème du point fixe, la plus petite solution est donnée par :
f_0 = Sup { Ω, τ(Ω), $\tau^2(\Omega)$, $\tau^3(\Omega)$, ….$\tau^n(\Omega)$, … }
Ω étant la fonction nul part définie.

Calculons d'abord τ(Ω) :
τ(Ω) = λn. SI (n > 100) Alors (n-10) Sinon Ω(Ω(n+11)) FSI
τ(Ω) = λn. SI (n > 100) Alors (n-10) FSI
Donc τ(Ω) = n-10 si n>100

Calculons $\tau^2(\Omega)$:
$\tau^2(\Omega) = \tau(\tau(\Omega))$ = λn. SI (n > 100) Alors (n-10) Sinon τ(Ω)(τ(Ω)(n+11)) FSI
posons x = τ(Ω)(n+11) et y = τ(Ω)(τ(Ω)(n+11)) = τ(Ω)(x)
D'après la définition de τ(Ω), x = (n+11)-10 si (n+11) > 100, c-a-d x = n+1 si n> 89
Donc y = τ(Ω)(x) = (n+1) – 10 si (n+1) > 100 et n > 89, c-a-d y = n-9 si n > 99
Comme on est dans le cas du SINON, donc n doit être ≤ 100.
De ce fait, y = n-9 si n > 99 et n ≤ 100, c-a-d y = 91 si n = 100.
Finalement, on aura alors :
$\tau^2(\Omega)$ = n-10 si n > 100 et $\tau^2(\Omega)$ = 91 si n = 100.

Calculons $\tau^3(\Omega)$:
$\tau^3(\Omega) = \tau(\tau^2(\Omega))$ = λn. SI (n > 100) Alors (n-10) Sinon $\tau^2(\Omega)$($\tau^2(\Omega)$(n+11)) FSI
posons x = $\tau^2(\Omega)$(n+11) et y = $\tau^2(\Omega)$($\tau^2(\Omega)$(n+11)) = $\tau^2(\Omega)$(x)
D'après la définition de $\tau^2(\Omega)$,
x = (n+11)-10 si (n+11) > 100 et x = 91 si n+11 = 100

c-a-d $x = n+1$ si $n > 89$ et $x = 91$ si $n = 89$
Donc $y = \tau^2(\Omega)(n+1)$ si $n>89$ et $y = \tau^2(\Omega)(91)$ si $n = 89$
c-a-d $y = n-9$ si $n>99$ et $n>89$ et $y = 91$ si $n=99$
Comme on est dans le cas du SINON, on rajoute donc la condition $n \leq 100$.
De ce fait, $y = n-9$ si $n > 99$ et $n>89$ et $n \leq 100$ et $y = 91$ si $n=99$ et $n \leq 100$
c-a-d $y = 91$ si $n = 100$ ou $n = 99$
Finalement, on aura alors :
$\tau^3(\Omega) = n-10$ si $n > 100$ et $\tau^3(\Omega) = 91$ si $n = 100$ ou $n=99$.

La forme générale (qui reste à démontrer par récurrence) est alors comme suit :
$\tau^k(\Omega) = n-10$ si $n > 100$ et $\tau^k(\Omega) = 91$ si $n = 100$ ou $n=99$ ou $n=98$ ou … $n = 100-k+2$.
C'est l'hypothèse de récurrence.

Montrons alors que :
$\tau^{k+1}(\Omega) = n-10$ si $n > 100$ et $\tau^{k+1}(\Omega) = 91$ si $n = 100$ ou $n=99$ ou $n=98$ ou … $n = 100-(k+1)+2$.
C-a-d :
$\tau^{k+1}(\Omega) = n-10$ si $n > 100$ et $\tau^{k+1}(\Omega) = 91$ si $n = 100$ ou $n=99$ ou $n=98$ ou … $n = 100-k+1$.

D'après le programme récursif, on peut écrire :
$\tau^{k+1}(\Omega) = \tau(\tau^k(\Omega)) = \lambda n.$ SI $(n > 100)$ Alors $(n-10)$ Sinon $\tau^k(\Omega)(\tau^k(\Omega)(n+11))$ FSI
Posons $x = \tau^k(\Omega)(n+11)$ et $y = \tau^k(\Omega)(x) = \tau^k(\Omega)(\tau^k(\Omega)(n+11))$

D'après l'hypothèse de récurrence, on peut écrire :
$x = (n+11) – 10$ si $(n+11) > 100$ et $x = 91$ si $(n+11) = 100$ ou $(n+1) = 99$ ou … $(n+11) = 100-k+2$
$x = n+1$ si $n>89$ et $x = 91$ si $n=89$ ou $n=88$ ou … ou $n = 89-k+2$
Dans ce cas, on aura alors :
$y = (n-9)$ si $n>99$ et
$y = 91$ si $n = 99$ ou $n = 98$ ou … $n = 100-k+1$ et
$y = \tau^k(\Omega)(91)$ si $n=89$ ou $n=88$ ou … ou $n = 89-k+2$
Comme on est dans le partie SINON, on rajoute la contrainte que $n \leq 100$, ce qui donne :
$y = 91$ si $n=100$ et
$y = 91$ si $n = 99$ ou $n = 98$ ou … $n = 100-k+1$ et
$y = \tau^k(\Omega)(91)$ si $n=89$ ou $n=88$ ou … ou $n = 89-k+2$

Finalement, on aura alors :
$\tau^{k+1}(\Omega) = n-10$ si $n > 100$ et $\tau^{k+1}(\Omega) = 91$ si $n = 100$ ou $n=99$ ou … $n = 100-k+1$

L'hypothèse de récurrence est donc correcte.

La fonction exactement calculée par ce programme récursif est la fonction **f91**définie par :
$f91 = \lambda n.$ SI $(n > 100)$ Alors $(n-10)$ Sinon 91 FSI