Université d'El-Oued
Département d'informatique
Chargée de module : Chourouk Guettas

Niveau : 3 LMD
Année : 2017-2018
Durée : 01:30 h

Examen de Génie Logiciel 2 – corrigé type

Questions de cours : (7 points)

1) Donner une définition du GL ?

   La science dont l'objectif est de mettre en place : des **outils**, **langages**, **méthodes**, **approches** favorisant la production de logiciels de qualité**. (1 point)**

2) Pourquoi le GL est considéré comme une science multidisciplinaire ? **(1 point)**

   Car elle n'exige pas uniquement la compétence de programmation, mais d'autre compétences: analyse (pour bien comprendre le problème à résoudre), gestion (pour gérer l'équipe de développement, le calendrier de réalisation, et aussi le budget offert), psychologie et sociologie (pour pouvoir interagir avec un client non informaticien, pour pouvoir coopérer avec des partenaires de travail, pour pouvoir gérer les problèmes de nature humaines qui peuvent se poser dans toute grande réalisation où des chantiers collaborent ensemble).

3) Citez deux avantages de l'approche des incréments. **(1 point)**

   - Bonne gestion de l'équipe (gain du temps)
   - Livraison parallèle

4) Pourquoi utilise-t-on les termes : Black-box, white-box dans l'activité de test logiciel ?

   On utilise le terme « White Box » pour indiquer que les données sont sélectionnées à partir de la structure de texte (boite transparente = code visible).

   Et le terme « Black Box » pour indiquer que les données sont sélectionnées à partir d'une spécification (code invisible) **(1 point)**

5) Enoncé :

   a) Le type de cet énoncé est objectif, Justification : les besoins sont mesurables. **(1 point)**
   b) Objectifs : Facile à utiliser => Durée nécessaire à apprendre le systtème ou toutes les fonctionnalités affichées **(1 point)**
      minimise les erreurs => nombre d'erreurs admises/ durée de travail ou nombre d'erreurs/H **(1 point)**

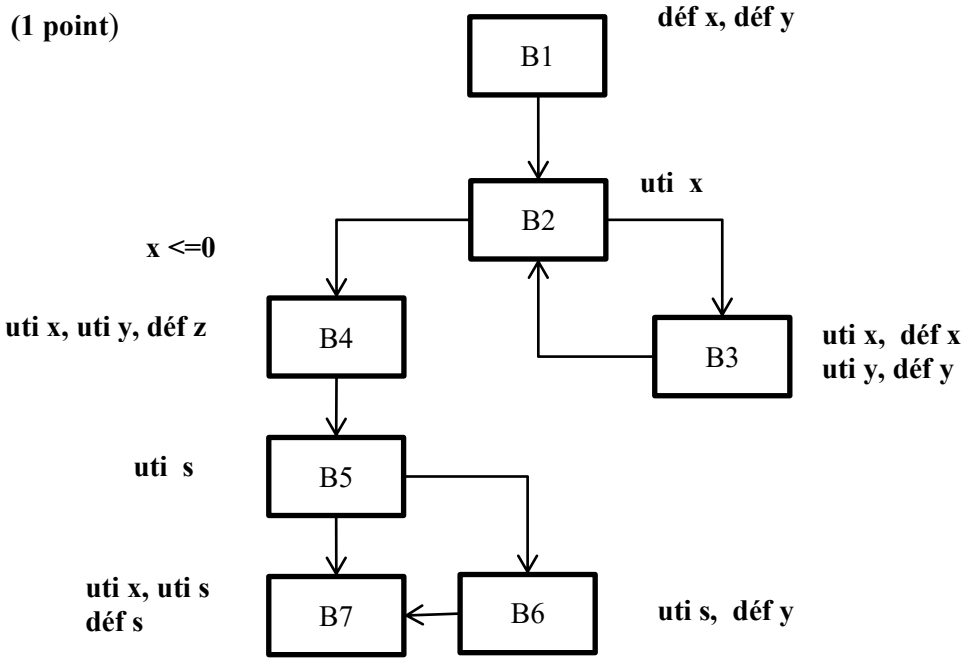
Exercice 1: (Activité de test) (7 points)

1) Soit le fragment du code suivant :

```
Read(x)
Read(y)
while (x > 0) do
Begin
        x = x - 1
        y = y +1
End
z = x + y
if (s > 0) then y = 3*s
s = x + s
```

a) Etablir un test pour détecter les anomalies concernant les variables **s** et **z** de ce code. Après cette analyse, proposer une correction dans laquelle le variable **z** sera remplacé par le variable **s.**

(1 point)



Les chemins possibles d'exécutions du programme (4 chemins possibles) (1 point)

C1: B1, B2, B4, B5, B7
C2: B1, B2, B4, B5, B6, B7
C3: B1, B2, B3, B2, B4, B5, B7
C4: B1, B2, B3, B2, B4, B5, B6, B7

variable s: (0.5 point)

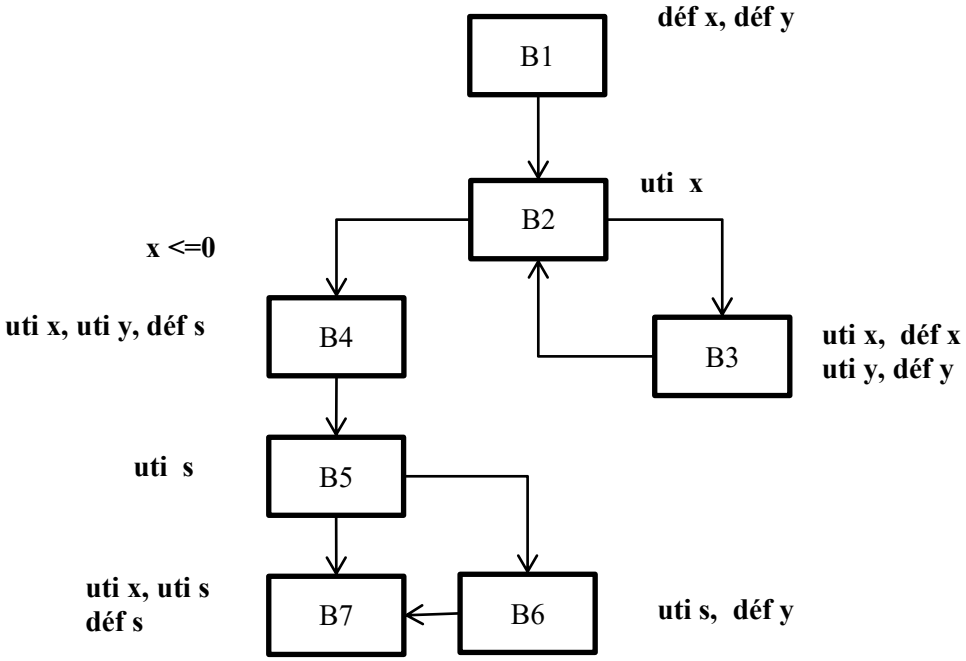
C1: uti s, uti s, déf s **(anomlie utilisation sans definition)**
C2: uti s, uti s, uti s, déf s **(anomlie utilisation sans definition)**
C3: uti s, uti s, déf s **(anomlie utilisation sans definition)**
C4 : uti s, uti s, uti s, déf s **(anomlie utilisation sans definition)**

variable z: (0.5 point)

C1: déf z **(anomlie definition sans utilisation)**
C2: déf z **(anomlie definition sans utilisation)**
C3: déf z **(anomlie definition sans utilisation)**
C4: déf z **(anomlie definition sans utilisation)**

Une correction dans laquelle le variable z sera remplacé par le variable s (0.5 point)

```
Read(x)
Read(y)
while (x > 0) do
Begin
        x = x - 1
        y = y +1
End
s = x + y
if (s > 0) then y = 3*s
s = x + s
```



b) Etablir un test dynamique avec deux couvertures (couverture de tous les enchaînements, couverture de toutes les utilisations du variable x).

Couverture de tous les enchaînements (autres couvertures sont possibles) (1 point)
Couverture = { B1, B2, B3, B2, B4, B5, B6, B7
B1, B2, B4, B5, B7}

Couverture de toutes les utilisations du variable x : (2 points)
Déf x (1)
C1 : B1, **B2**, B3 ....
C2: B1, **B2**, B4 …
C3: B1, B2, **B3**, B2, …
C4: B1, B2, … **B4**
C5: B1, B2, … **B7**

➡ {B1**, B2, B3,** B2, B4 **… B7**}

Déf x (2)
C1 : B1, B2, B3, **B2**, ...
C2: B1, B2, B3, B2,... **B4**
C3: B1, B2, B3, B2,... **B7**

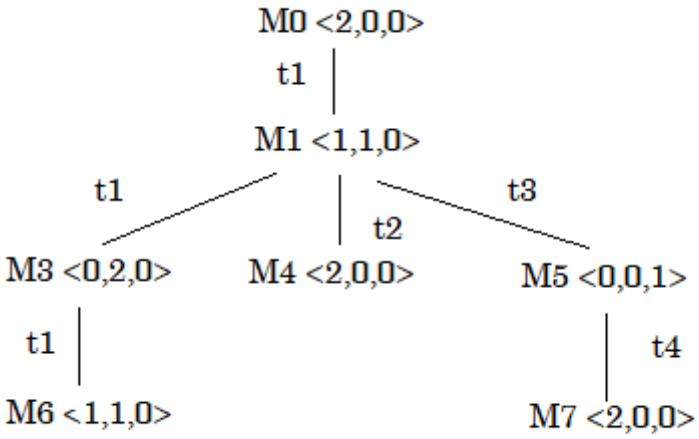
➡ {B1, B2, B3, **B2**, … **B7**}

c) Créer un mutant de ce fragment. **(0.5 point)**

```
Read(x)
Read(y)
while (x < 0) do
Begin
        x = x - 1
        y = y +1
End
s = x + y
if (s > 0) then y = 3*s
s = x + s
```

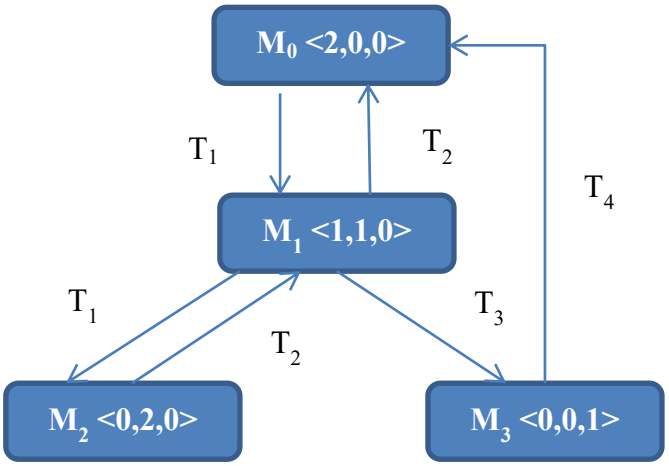
Exercice 2: (Réseaux de Petri) (6 points)

1) Tracer L'arbre de marquages atteignables : **(1.5 points)**



- Ce RdP est Borné et sans blocage. **(0.5 point)**

2) Proposer un graphe pour l'arbre obtenu. **(1.5 points)**



3) Soit $\delta = <T_1, T_1, T_2, T_3, T_4>$ Trouver le marquage M obtenu en franchissant δ depuis M_0 (algébriquement). **(1.5 points)**

$$\mathbf{M^t = M_0^t + WV_\delta^t} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 1 & -1 & 2 \\ 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \mathbf{x} \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

4) À partir du marquage initial, on aboutit à un marquage final identique au marquage initial
➡ RdP réversible **(1 points)**