Examen de "Test Logiciel" 2009-2010

Consignes :

- Le partiel dure deux heures.
 - Les documents ne sont pas autorisés.
 - Les exercices valent à peu près tous 5 points.
 - Des réponses concises et précises sont à privilégier.
-

Exercice 1 (Petites questions de cours)

1. Citez les différentes phases du cycle en V avec les types de test correspondants.
2. Citez les différentes phases du processus de test. Pour chaque phase, dire si elle est facilement automatisable ou non (sans donner plus de détails).
3. Citez deux avantages et deux défauts du test aléatoire uniforme.
4. Citez 4 cas où l'oracle est automatisable. Précisez si c'est un oracle partiel ou parfait.
5. Tests de régression : citez 5 problèmes liés aux tests de régression. Quels problèmes JUnit permet de résoudre ?

Exercice 2 (Exécution symbolique)

1. Qu'est qu'un prédicat de chemin ? Expliquez l'intérêt pour le test.
2. Donnez l'algorithme d'exécution symbolique basique (récuratif) pour un CFG dont les instructions sont : des blocs d'instructions atomiques *block instructions* : *node-next*, des conditions *ite(condition, node-if, node-then)* ou un noeud d'arrêt *stop*. L'algorithme pourra utiliser une fonction de traduction *trad* (ne pas l'écrire) des instructions et conditions vers le langage de formule.
Sortie de l'algorithme : mise à jour d'une variable globale contenant les couples (DT, chemin suivi) trouvés.
3. Définissez les notions de correction et de complétude pour l'exécution symbolique (vis à vis de la génération de tests). Quelles conditions faut-il pour que votre procédure soit correcte ? complète ?
4. Qu'est-ce que l'exécution concolique ?
5. Montrez deux problèmes de l'exécution symbolique. Expliquez comment l'exécution concolique permet d'y remédier.

Exercice 3 (Test boîte noire) Un programme prend en entrée trois flottants, interprétés comme les longueurs des côtés d'un triangle. Le programme répond s'il s'agit d'un triangle scalène (= triangle quelconque), isocèle ou équilatéral. On suppose que l'en-tête de la fonction est : `kind my-prog(float a, float b, float c)`, où `kind` est un type énuméré pouvant valoir `SCAL`, `ISOC`, `EQUIL`.

fonctionnement normal avec autres caractères
→

1. Produire une suite de tests nominaux (fonctionnement normal) pour ce programme : CT et oracle.
2. Produire des tests aux limites (robustesse) pour ce programme (ne faire que des tests qui respectent le type et le nombre d'arguments). → :lem

Exercice 4 (Test en boîte blanche) Soit le programme C suivant :

```
1 char f, o, r, v1, v2, v3 ;
2 f = getchar(); /* input a partir d'un fichier */
3 o = getchar(); /* input du prochain caractere */
4 r = getchar();
5 if (f != 'F' || o != 'O' || r != 'R') goto error;
6 v1 = getchar();
7 if (v1 != '_') goto error;
8 v1 = getchar();
9 if (isalpha(v1)) { /*v1 est-il alphanumerique (minuscules, majuscules, chiffres)?*/
10 v2 = getchar();
11 v3 = getchar();
12 if (v2 == '_' || (isalpha(v2) && v3 == '_'))
13 goto end;
14 else
15 goto error;
16 }
17 else
18 goto error;
19 end:
20 return 1;
21 error:
22 return 0;
```

1. Donner le graphe de contrôle du programme.
2. Donner une suite de tests minimale TS_n (ici suite des lectures de caractères) qui couvre tous les noeuds du graphe de contrôle, mais pas toutes les branches (aussi appelées : arcs, décisions). Justifier votre réponse en précisant les noeuds/branches couverts par chaque test, et les branches non couvertes au final. (remarque : suite de tests minimale : si on en enlève un, la couverture n'est plus assurée).
3. Ajouter des données de test pour obtenir une suite de tests TS_a qui couvre tous les arcs.
4. Donnez les def, les c-use et les p-use de la variable v1.
5. Que sont les critères all-def, all-use, all-def-use (ou all-DU) ?
6. Définissez les critères I, D (décisions / branches), C (conditions), DC, MC/DC, MC (vous pouvez illustrer par un exemple).
7. Montrez que $C < DC$, $D < DC$, $DC < MC/DC$, $MC/DC < MC$. (sur un ou des exemples, montrez des tests qui résolvent l'un mais pas l'autre).