TD de Test Logiciel

Sélection de tests boîte blanche

Exercice 1 (CFG, DFG). *Écrivez le graphe de contrôle de l'algorithme suivant :*

```
1  void my_fun(int a, int b, int c, int x) {
2   if (b < c) {
3     d = 2*b;
4     f = 3*c;
5     if (x >= 0 && a >=0) {
6       d = x;
7       e = c;
8       if (d==0) {
9         f = f-e;
10        if (d < a)
11            d=a+1
12        else
13            d=a-1;
14        write(a);
15      } else exit(0);
16    } else exit(0);
17  } else exit(0);
18 }
```

- *Donner les noeuds du CFG. Donner les instructions branchantes, à chaque instruction branchante dire ses décisions / branches et ses conditions.*
- *Donner les chemins de ce graphe de contrôle.*
- *Pour chaque variable, donner l'ensemble de ces définitions.*
- *Pour chaque variable, donner l'ensemble de ces utilisations. Préciser si ce sont des c-use ou des p-use.*
- *Donner l'ensemble des paires def-use pour chaque variable du programme.*

Exercice 2 (Critères de couverture orientés contrôle). *Soit le programme ci-dessous.*

```
1
2  void foo(bool a, bool b, bool c) {
3    if (a or (b and c)) then
4       println(``ok'');
5    else
6       ();
7    println("fin");
8  }
```

(1) Pour chacun des critères ci-dessous, donner les éléments à couvrir : instructions (I), décisions (D), conditions (C), décisions/conditions (DC), conditions multiples (MC), MC/DC.

(2) Donner des jeux de tests du programme couvrant les critères et montrant qu'ils sont différents.

Exercice 3 (Critères de couverture orientés donnés)**.** *Reprendre le programme de l'exercice 1.*

(1) Pour chacun des critères ci-dessous, donner les éléments à couvrir : all-defs, all-use, all-p-use, all-c-use, all-def-use-paths.

(2) Donner des jeux de tests du programme couvrant les critères et montrant qu'ils sont différents.

Exercice 4 (Couverture structurelle)**.** *Soit le programme C suivant :*

```
1  /* Outputs result = 0+1+...+|value|
2   * if result > maxint then error
3   */
4  void maxsum(int maxint, int value) {
5    int result =0;
6    int i =0;
7    if (value < 0)
8      value = -value;
9    while (i< value && result <= maxint) {
10     i++;
11     result = result+i;
12   }
13   if (result <= maxint)
14       println(result);
15   else
16       println("error");
17 }
```

1. *Donner le graphe de contrôle du programme.*
2. *Donner une suite de tests TS_n qui couvre tous les noeuds du graphe de contrôle. Justifier votre réponse.*
3. *La suite TS_n couvre-t-elle tous les arcs ? Si oui, indiquer les données de tests qui effectuent la couverture des arcs. Sinon, ajouter des données de test pour obtenir une suite de tests TS_a qui couvre tous les arcs.*
4. *Indiquer quelles sont les lignes de code correspondant aux définitions de la variable* `result` *(ensemble defs(*`result`*)). Pareil pour les ensembles d'utilisation en calcul c-use(*`result`*) et d'utilisation en prédicats p-use(*`result`*).*
5. *Donner une suite de test TS_d qui couvre le critère all-use-one-def pour les p-use de* `result`*.*
6. *Explicitez les chemins à couvrir pour le critère all-use-all-def pour les p-use de* `result`*.*

Exercice 5 (Mutations)**.**

- *Prendre le programme de l'exercice précédent, générer 5 mutants (ordre 1) distinguables au moyen de ROR et ABS.*
- *Générer les DT pour tuer ces mutants, calculer le score de mutation obtenu, calculer les couvertures I, C, D obtenues.*
- *Prenez les DT que votre voisin a trouvé pour la question précédente, calculer le score de mutation de ces DTs sur vos mutants.*
- *Générer 1 mutant non distinguable avec ROR et 1 mutant non distinguable avec ABS.*
- *Reprendre chacun de vos jeux de tests de l'exo précédent, et calculer son score de mutation.*
- *Définissez un opérateur de mutations O tel que un jeu de tests O-adéquat couvre toutes les instructions. Idem avec décisions et conditions.*