Universität des Saarlandes
Lehrstuhl für Softwaretechnik
Prof. Andreas Zeller
Valentin Dallmeier, Silvia Breu, Stephan Neuhaus

13. Übung Softwaretechnik

Testverfahren

Abgabe: Freitag 10.02.06, 12 Uhr

1 Kontrollflussorientiertes Testen (18 Punkte)

Die folgende Methode implementiert den euklidischen Algorithmus zur Berechnung des grössten gemeinsamen Teilers positiver Zahlen in C:

```
int ggt(int a, int b) {
  int tmp, remainder;

  if((a<=0) || (b<=0)) {
    return -1;
  }
  if (a < b) {
    tmp = a;
    a = b;
    b = tmp;
  }
  remainder = a % b;
  while (remainder != 0) {
    a = b;
    b = remainder;
    remainder = a % b;
  }
  return b;
}
```

Für den Fall, dass mindestens eines der beiden Argumente kleiner als 1 ist (erstes `if`), wird -1 zurückgegeben. Falls notwendig, werden die beiden Argumente vertauscht (zweites `if`). Nach der Berechnung in der `while`-Schleife wird das Ergebnis ausgegeben.

a) Wandeln Sie das Programm in einen Kontrollflussgraphen gemäß dem in der Vorlesung vorgestellten Beispiel um. Achten Sie dabei darauf, dass Ihr Graph Start-und Endknoten hat. Gehen Sie davon aus, dass im Code dieser Methode keine weiteren Exceptions ausgelöst werden können. Jede `if` und `while` Anweisung soll durch einen Knoten abgebildet werden. (6 Punkte)

b) Geben Sie eine Menge von Testfällen an, also Belegungen für die Parameter a und b, so dass die Tests Anweisungsüberdeckung sicherstellen. Geben Sie für jeden Test an, wie der Kontrollfluss im Testlauf aussieht, d.h. ob und ggf. wie oft `if` oder `while` Schleifen durchlaufen werden. Dies gilt für alle Testfälle in Aufgabe 1. (3 Punkte)

c) Wie unterscheiden sich Anweisungsüberdeckung und Zweigüberdeckung? Begründen Sie, ob die in Aufgabe b) angegebenen Testfälle bereits Zweigüberdeckung realisieren. Falls nein, geben sie weitere Testfälle an, so dass Zweigüberdeckung besteht. (2 Punkte)

d) Welches grundsätzliche Problem besteht, wenn man durch Testfälle Pfadüberdeckung im Kontrollflussgraphen erreichen möchte? (1 Punkt)

e) Wie unterscheiden sich Boundary-Interior-Pfadtest und Pfadüberdeckung? Begründen Sie, ob Ihre bisher angegebenen Testfälle bereits dem Boundary-Interior-Pfadtest genügen. Falls nicht, geben Sie weitere Testfälle an, so dass das Kriterium erfüllt ist. (2 Punkte)

f) Schreiben Sie ein Programm in einer Datei `ggt.c`, das die obige Methode enthält, und zusätzlich noch folgende `main` Methode:

```
int main() {
  printf("ggt(11324,23443)=%d\n",ggt(11324,23443));
}
```

Zusätzlich müssen sie noch die Anweisung `#include <stdio.h>` einfügen. Übersetzen Sie anschliessend mit dem in der Vorlesung angegebenen Befehl `gcc -g -fprofile-arcs -ftest-coverage -o ggt ggt.c` das Program und führen Sie es aus. Starten Sie gcov mit dem Befehl `gcov ggt`. Die Ausgabe wird in der Datei `ggt.c.gcov` abgespeichert. Geben Sie einen Ausdruck dieser Datei ab. Wie oft wird die Schleife zur Berechnung des ggt durchlaufen? (4 Punkte)

2 Minimale Mehrfach-Bedingungsüberdeckung (6 Punkte)

Nach erfolgreichem Abschluss Ihres Studiums arbeiten Sie bei der Firma, die mit der Realisierung des aus der Klausur bekannten Parkplatzbewirtschaftungssystems betraut ist. Dort finden Sie folgendes Java-Fragment, das ausgeführt wird, wenn eine Karte in das System eingebucht werden soll:

```
public boolean login(User user) throws NotAuthorizedException{
  if ((!isAlreadyLoggedIn(user) && user.hasNeededCash()) ||
       user.isSuperUser) {
    ... perform login ...
  } else {
    throw new NotAuthorizedException();
  }
}
```

Ein Benutzer darf eingebucht werden, wenn er nicht bereits eingebucht ist und genügend Geld hat, oder wenn er Superuser ist (z.B. Hausmeister).

a) Erläutern Sie anhand eines selbstgewählten Beispiels, warum es bei geforderter Mehrfach-Bedingungsüberdeckung Schwierigkeiten geben kann, einen Testplan zu erstellen? (2 Punkte)

b) Erstellen Sie einen Testplan für alle Situationen die notwendig sind, damit für den obigen Code minimale Mehrfach-Bedingungsüberdeckung erreicht wird. (4 Punkte)

3 Mutationstesten (6 Punkte)

a) Für welche Zwecke kann Mutationstesten eingesetzt werden? (2 Punkte)

b) Um die Güte Ihrer Testsuite zu messen, führen Sie 2000 Mutationen in Ihr Programm ein, wovon 1000 durch die Testsuite aufgedeckt werden. Was schließen Sie daraus? (2 Punkte)

c) Aufgrund des Ergebnisses von Aufgabe b) überarbeiten Sie Ihre Testsuite und finden von 2000 Mutationen 1950 wieder. Zusätzlich finden Sie 8 neue Fehler im Programm. Was schliessen Sie daraus über die Zahl der Restfehler im Programm? (2 Punkte)

Abgabe

Bilden Sie Teams aus je zwei Studenten, erarbeiten Sie die Lösung *gemeinsam* und reichen Sie *eine Lösung pro Team* ein. Drucken Sie Ihre Lösungen aus, klammern Sie sie zusammen und werfen Sie sie in die mit “Softwaretechnik“ beschrifteten Übungskästen vor Hörsaal 1 in Gebäude E1 1 (45). Abgaben per E-mail werden nicht akzeptiert.

Fragen?

Fragen zu diesem Übungsblatt können sie in Ihrer Übungsgruppe stellen.