Institut für Informatik
Jun.-Prof. Dr.-Ing. Elke Pulvermüller
B. Sc. Sergey Krutikov
B. Sc. Alexander Brandt

Universität Osnabrück, 20.11.2012
`http://www-lehre.inf.uos.de/~sq`

Testat bis Freitag, 30.11.2012, 19:00 Uhr

Übungen zu Software-Qualität

Wintersemester 2012/2013

Übungsblatt 6

Aufgabe 6.1 (LCSAJ-Test/*Linear Code Sequence And Jump*, 15 Punkte)

Machen Sie sich mit der Methode `pow()` auf der Rückseite des Aufgabenblattes vertraut. Sie finden die Methode auch in in der JAVA-Klasse unter der folgenden Internet-Adresse:

`http://www-lehre.inf.uos.de/~sq/2012/aufgaben/blatt06/Pow.java`

a) Bestimmen Sie die existierenden LCSAJs in der Methode `pow()`. Markieren Sie sie *grafisch* eindeutig und notieren Sie sie in Tripel-Schreibweise. (11 Punkte)

b) Bilden Sie (möglichst wenige) Testfälle, so dass alle LCSAJs abgedeckt werden. Machen Sie dabei kenntlich, welcher Testfall welche LCSAJs abdeckt. Begründen Sie, wenn sich ein Testfall für einen LCSAJ nicht erzeugen lässt. (4 Punkte)

Aufgabe 6.2 (McCabe-Überdeckungstest/*Basic-Path-Test*, 15 Punkte)

Diese Aufgabe bezieht sich ebenfalls auf die Methode `pow()`.

a) Erzeugen Sie den Kontrollflussgraphen der Methode `pow()`. (2 Punkte)

b) Berechnen Sie die zyklomatische Komplexität von `pow()` und geben Sie Ihren Rechenweg nachvollziehbar an. (1 Punkt)

c) Was sagt diese Zahl allgemein aus? (2 Punkt)

d) Geben Sie die Elementarpfade von `pow()` sowohl als Folge von Knoten als auch in Vektorschreibweise (Kanten) an. (4 Punkte)

e) Erzeugen Sie mit Hilfe der Elementarpfade eine Linearkombination, so dass sich der Pfad für den Testfall 2^2 ($x = 2, n = 2$) erzeugen lässt. (2 Punkt)

f) Geben Sie, um das Überdeckungskriterium nach McCabe zu erreichen, für jeden Elementarpfad einen entsprechenden Testfall an. Machen Sie dabei kenntlich, welcher Testfall welchen Elementarpfad abdeckt. Begründen Sie, wenn sich ein Testfall nicht erzeugen lässt. (4 Punkte)

Aufgabe 6.3 (Offener Frageteil, 10 Punkte)

Beantworten Sie Ihrer Tutorin bzw. Ihrem Tutor Fragen zur Veranstaltung „Software-Qualität“.

– bitte wenden –

```
01    /**
02     * Berechnung der n-ten Potenz.
03     *
04     * @param x Basis
05     * @param n Exponent
06     * @return n-te Potenz zur Basis x
07     */
08    public static double pow(double x, int n) {
09
10        double res = 1.0;
11        int i;
12
13        if (n < 0) {
14            i = -n;
15        } else {
16            i = n;
17        }
18
19        while (i > 0) {
20            res *= x;
21            i--;
22        }
23
24        if (n < 0) {
25            res = 1 / res;
26        }
27
28        return res;
29    }
```