Thema, Ziele: Verifikation und Validation (V&V)

Aufgabe 1: Codeabdeckung (code coverage)

Gegeben ist der unten stehende Codeausschnitt mit den drei Integervariablen a, b und c.

```
if (a>4)
{
  if (b==7)
  {  // do sth
  }
  else
  {
    if (c>=9)
    { // do sth
    }
  }
}
else
{
  if (b>7)
  { // do sth
  }
  if (c<4)
  { // do sth
  }
}
```

a) Zeichnen Sie den Verzweigungsbaum graphisch auf, d.h. jede Anweisung (im vorliegenden Code häufig ein leerer Block) wird ein Knoten, jede Verzweigung ein Zweig.

b) Definieren Sie die Wertepaare der Variablen a, b und c (Testfälle), damit Sie Anweisungsüberdeckung (instruction coverage) erreichen.

c) Definieren Sie die Wertepaare der Variablen a, b und c (Testfälle), damit Sie Zweigüberdeckung (branch coverage) erreichen.

d) Definieren Sie die Wertepaare der Variablen a, b und c (Testfälle), damit Sie Pfadüberdeckung (path coverage) erreichen.

Hinweise:

- Bei Anweisungsüberdeckung muss jeder Knoten im Verzweigungsbaum mindestens einmal besucht werden.
- Bei Zweigüberdeckung muss jeder Zweig im Verzweigungsbaum mindestens einmal besucht werden.
- Bei Pfadüberdeckung müssen alle Pfade, d.h. alle Zweigkombinationen im Verzweigungsbaum besucht werden.

Aufgabe 2: Primzahlen bestimmen (Prüfungsaufgabe im FS 2011)

Die Funktion `isPrime()` soll überprüfen, ob eine Zahl `cand` eine Primzahl ist oder nicht.

```
bool isPrime(int cand)
{
  int b = 2;
  bool p = true;
  while (p && (b * b < cand))
  {
    if (cand % b == 0)
      p = false;
    b++;
  }
  return p;
}
```

a) (2 Punkte) Bestimmen Sie den Aufwand von `isPrime()` in O-Notation (mit Begründung).

b) (4 Punkte) Definieren Sie geeignete Äquivalenzklassen für den Test dieser Funktion.

c) (4 Punkte) Definieren Sie geeignete Testfälle.

d) (4 Punkte) Ermitteln Sie mittels statischer Codeanalyse (Code Inspection) für jeden Testfall aus Aufgabe c) den wirklichen Ausgabewert, den die obenstehende Funktion liefert.

e) (1 Punkt) Funktioniert `isPrime()` fehlerfrei? Begründen Sie Ihre Aussage.

Hinweis: Sie müssen den Algorithmus nicht korrigieren, falls er nicht fehlerfrei arbeitet.

Aufgabe 3: Implementation des Primzahlenalgorithmus

a) Implementieren Sie die fehlerhafte Funktion `isPrime()` aus Aufgabe 3. Implementieren Sie dazu auch ein Testprogramm, welches die definierten Testfälle ausführt. Vergleichen Sie die Ausgabe mit Ihren theoretisch ermittelten Werten.

b) Korrigieren Sie die Funktion `isPrime()` aus Aufgabe 3. Das Testprogramm aus Teilaufgabe a) sollen Sie dazu unverändert lassen. Überprüfen Sie, ob Ihre Funktion nun die korrekten Ergebnisse liefert.