Gedächtnisprotokoll – Test01A 16.01.26

Aufgabe 1 – Multiple Choice
_/3 P

Zu jeder Frage gibt es genau **eine richtige Antwort**.
Es gibt **keinen Punktabzug** für falsche Antworten.
Werden **mehr als eine Antwort** angekreuzt, wird die Frage mit **0 Punkten** bewertet.

a) Welche Testart prüft primär Schnittstellen zwischen Komponenten?
_/0.5 P

[ ] Modultest
[X] Integrationstest
[ ] Regressionstest
[ ] Abnahmetest

b) Was ist ein typischer Code-Smell?
_/0.5 P

[ ] Kurze Methoden
[X] Lange Methoden oder große Klassen
[ ] Hohe Kohäsion
[ ] Niedrige Kopplung

c) Was ist ein wesentliches Merkmal von Refactoring
_/0.5 P

[ ] Funktionalität wird erweitert
[X] Das beobachtete Verhalten bleibt erhalten
[] Fehler werden bewusst in Kauf genommen
[] Tests werden überflüssig

d) Welche Arten von Metriken werden in der Softwaretechnik unterschieden?
_/0.5 P

[X] Produktmetriken und Prozessmetriken
[ ] Additive und multiplikative Metriken
[ ] Deterministische und nicht-deterministische Metriken
[ ] Statistische und qualitative Prozessmetriken

e) Für welche Projekte eignet sich das Wasserfallmodell besonders?
_/0.5 P

[] Kleine Start-up-Projekte mit unsicheren Anforderungen
[] Forschungsprojekte mit hohem Innovationsgrad
[] Projekte mit häufig wechselnden Kundenwünschen
[X] Große Projekte mit stabilen und klaren Anforderungen

f) Welche Testarten sind im V-Modell vorgesehen?
_/0.5 P

[] Nur Modultests
[] Nur Abnahme- und Systemtests
[X] Komponenten-, System- und Abnahmetests
[] Nur Komponenten- und Abnahmetests

Aufgabe 2 – Testen und Metriken
_/7 P

a) Halstead-Metriken
_/2 P

Gegeben ist folgendes Programmfragment:

```
double cube(double x) {
    return x * x * x;
}
```

Berechnen Sie die folgenden Halstead-Metriken:

Vokabular n = 8
_/0.5 P
Länge N = 13
_/0.5 P
Volumen V = 13 * log(8) = 39
_/0.5 P
Schwierigkeit D = 3 * 5/2 = 7,5
_/0.5 P

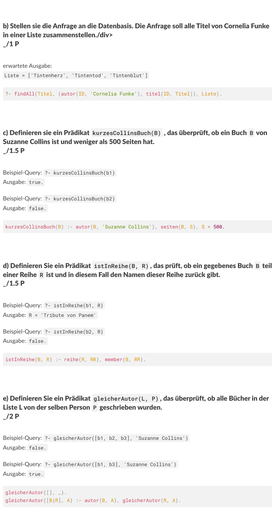
b) Kontrollflussgraph (CFG)
_/5 P

Gegeben ist das folgende Java-Programm:

```
01 int berechneWert(int x, int n) {
02     int count = 0;
03     if (x > 0 &&
04         n > 0) {
05         while (x % 2 == 0) {
06             x = x / 2;
07             count++;
08         }
09     } else {
10         return 0;
11     }
12     if (count > 3) {
13         x = 0;
14     } else {
15         x = 1;
16     }
17     return 2 * count + n + x;
18 }
```

Aufgabe:

Zeichnen Sie den vollständigen Kontrollflussgraphen (CFG) des Programms. Beschriften Sie die Knoten mit der zugehörigen Zeilennummer. Sie dürfen Knoten zusammenfassen, wenn aus dem Namen des Knoten eindeutig hervorgeht, welche Zeilen er zusammenfasst. Zeilen, die ausschließlich geschweifte Klammern oder ein `else` enthalten, werden nicht als eigene Knoten gezählt.



Aufgabe 3 – Prolog
_/10 P

Gegebene Datenbank

```
% autor(Person, Buch).
autor(b1, 'Cornelia Funke').
autor(b2, 'Cornelia Funke').
autor(b3, 'Cornelia Funke').
autor(b4, 'Suzanne Collins').
autor(b5, 'Suzanne Collins').

% titel(Buch, Titel).
titel(b1, 'Tintenherz').
titel(b2, 'Tintenblut').
titel(b3, 'Tintentod').
titel(b4, 'Die Tribute von Panem').
titel(b5, 'Catching Fire').

% seiten(Buch, Seitenzahl).
seiten(b1, 640).
seiten(b2, 640).
seiten(b3, 720).
seiten(b4, 384).
seiten(b5, 391).

% bewertung(Buch, Wert).
bewertung(b1, 8).
bewertung(b2, 9).
bewertung(b3, 7).
bewertung(b4, 9).
bewertung(b5, 8).

% reihe(Reihentitel, Buchliste).
reihe('Tintenherz', [b1, b2, b3]).
reihe('Panem', [b4, b5]).
```

a) Stellen sie eine Anfrage an die Datenbasis. Was ist die Bewertung von Tintenblut?
_/0.5 P

erwartete Ausgabe:
`X = 8`

```
?- bewertung(b10, X).
```

b) Stellen sie die Anfrage an die Datenbasis. Die Anfrage soll alle Titel von Cornelia Funke in einer Liste zusammenstellen./div>
_/1 P

erwartete Ausgabe:
`Liste = ['Tintenherz', 'Tintentod', 'Tintenblut']`

```
?- findall(Titel, (autor(ID, 'Cornelia Funke'), titel(ID, Titel)), Liste).
```

c) Definieren sie ein Prädikat `kurzesCollinsBuch(B)` , das überprüft, ob ein Buch `B` von Suzanne Collins ist und weniger als 500 Seiten hat.
_/1.5 P

Beispiel-Query: `?- kurzesCollinsBuch(b1).`
Ausgabe: `true.`

Beispiel-Query: `?- kurzesCollinsBuch(b2).`
Ausgabe: `false.`

```
kurzesCollinsBuch(B) :- autor(B, 'Suzanne Collins'), seiten(B, S), S < 500.
```

d) Definieren Sie ein Prädikat `istInReihe(B, R)` , das prüft, ob ein gegebenes Buch `B` teil einer Reihe `R` ist und in diesem Fall den Namen dieser Reihe zurück gibt.
_/1.5 P

Beispiel-Query: `?- istInReihe(b1, R).`
Ausgabe: `R = 'Tintenherz'.`

Beispiel-Query: `?- istInReihe(b2, R).`
Ausgabe: `false.`

```
istInReihe(B, R) :- reihe(R, BB), member(B, BB).
```

e) Definieren Sie ein Prädikat `gleicherAutor(L, P)` , das überprüft, ob alle Bücher in der Liste `L` von der selben Person `P` geschrieben wurden.
_/2 P

Beispiel-Query: `?- gleicherAutor([b1, b2, b3], 'Suzanne Collins').`
Ausgabe: `false.`

Beispiel-Query: `?- gleicherAutor([b4, b5], 'Suzanne Collins').`
Ausgabe: `true.`

```
gleicherAutor([], _).
gleicherAutor([B|R], A) :- autor(B, A), gleicherAutor(R, A).
```

f) Definieren Sie ein Prädikat `durchschnittBewertung(B, D)` , das die durchschnittliche Bewertung `D` aller Bücher in einer Liste `B` berechnet.
_/3.5 P

Beispiel-Query: `?- durchschnittBewertung([b1,b2,b3], D).`
Ausgabe: `D = 7.333333333333`

Beispiel-Query: `?- durchschnittBewertung([b3,b5,b9], D).`
Ausgabe: `D = 5.666666666667`

```
?- durchschnittBewertung(Bucher, Durchschnitt) :- summe_und_anzahl(Bucher, Summe, Anzahl),
summe_und_anzahl([], 0, 0).
summe_und_anzahl([B|Rest], Summe, Anzahl) :- bewertung(B, R),
summe_und_anzahl(Rest, SummeRest, AnzahlRest), Summe ist SummeRest + R, Anzahl ist AnzahlRes
```