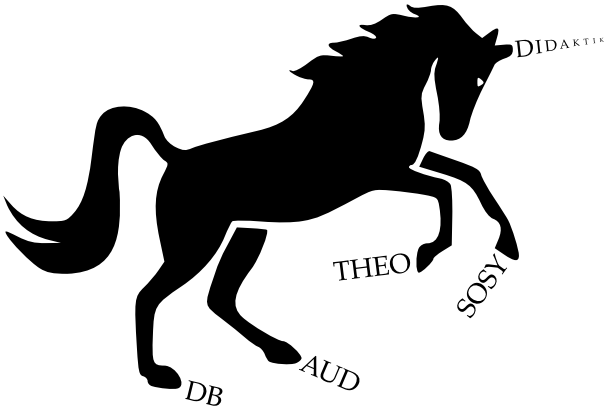
46116
Herbst
2015

Softwaretechnologie / Datenbanksysteme (nicht vertieft)

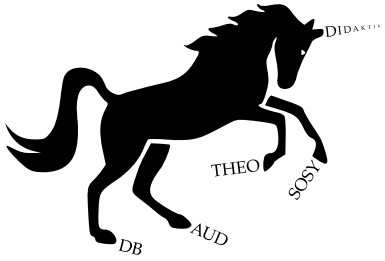
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Thema Nr. 1

Teilaufgabe Nr. 1

3. Projektmanagement [Gantt und PERT]

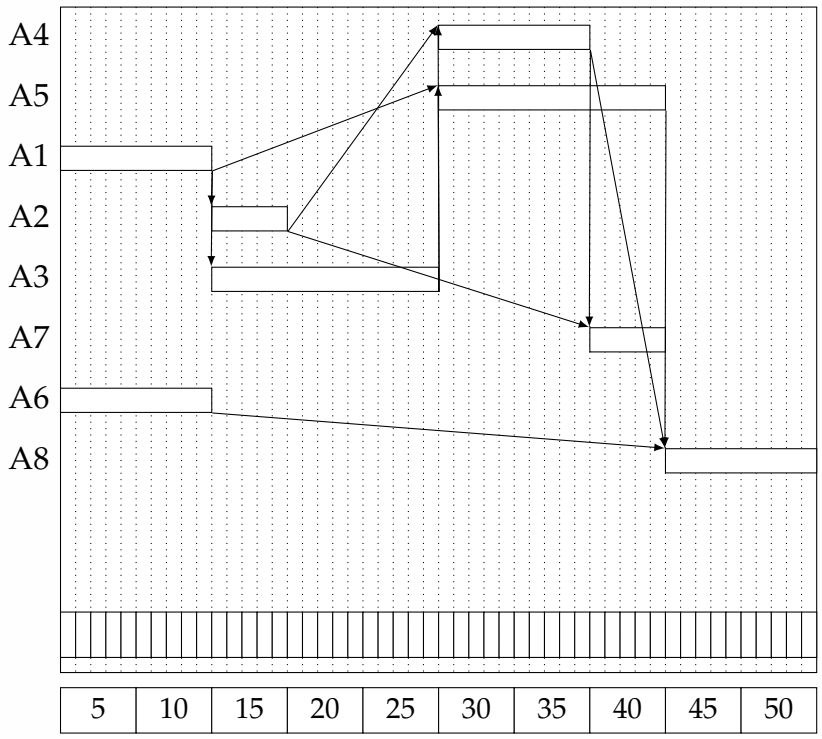
Betrachten Sie die folgende Tabelle zum Projektmanagement:

Name	Dauer (Tage)	Abhängig von
A1	10	
A2	5	A1
A3	15	A1
A4	10	A2, A3
A5	15	A1, A3
A6	10	
A7	5	A2, A4
A8	10	A4, A5, A6

Tabelle 1: Übersicht Arbeitspakete

(a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.

Lösungsvorschlag



(b) Wie lange dauert das Projekt mindestens?

(c) Geben Sie den oder die kritischen Pfad(e) an.

(d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

Teilaufgabe Nr. 2

Aufgabe 1 [Theoriefragen Datenbank]

Erläutern Sie die folgenden Begriffe in knappen Worten:

(a) Datenunabhängigkeit

Lösungsvorschlag

Änderungen an der physischen Speicher- oder der Zugriffsstruktur (beispielsweise durch das Anlegen einer Indexstruktur) haben keine Auswirkungen auf die logische Struktur der Datenbasis, also auf das Datenbankschema.

(b) Superschlüssel

Lösungsvorschlag

Ein Superschlüssel ist ein Attribut oder Attributkombination, von der alle Attribute einer Relation funktional abhängen.

(c) Referentielle Integrität

Unter Referentieller Integrität verstehen wir Bedingungen, die zur Sicherung der Datenintegrität bei Nutzung relationaler Datenbanken beitragen können. Demnach dürfen Datensätze über ihre Fremdschlüssel nur auf existierende Datensätze verweisen.

Danach besteht die Referentieller Integrität grundsätzlich aus zwei Teilen:

(i) Ein neuer Datensatz mit einem Fremdschlüssel kann nur dann in einer Tabelle eingefügt werden, wenn in der referenzierten Tabelle ein Datensatz mit entsprechendem Wert im Primärschlüssel oder einem eindeutigen Alternativschlüssel existiert.

(ii) Eine Datensatzlöschung oder Änderung des Schlüssels in einem Primär-Datensatz ist nur möglich, wenn zu diesem Datensatz keine abhängigen Datensätze in Beziehung stehen.

Aufgabe 4 [Versicherungsgesellschaft „Insurance Pro“]

Das Datenbankschema der Versicherungsgesellschaft „Insurance Pro“ enthält sowohl Kunden- als auch Mitarbeiterdaten, sowie Informationen über Schadensfälle. Das Schema ist von folgender Gestalt, wobei FS für Fremdschlüssel und PS für Primärschlüssel steht:

`Versicherungsnehmer (KNr, Name, Anschrift, Geburtsdatum)`

`Fahrzeug (Kennzeichen, Farbe, Fahrzeugtyp)`

`Mitarbeiter (MNr, Name, Kontaktdaten, Abteilung, Leiter)`

`Abteilung (ANr, Bezeichnung, Ort)`

`Versicherungsvertrag (VNr, Abschlussdatum, Art, Versicherungsnehmer, Fahrzeug, Mitarbeiter)`

`Schadensfall (SNr, Datum, Gesamtschaden, Beschreibung, Mitarbeiter)`

`Beteiligung (Schadensfall, Fahrzeug, Fahrzeugschaden)`

- *Abteilung* ist FS und bezieht sich auf den PS von *Abteilung*.
- *Versicherungsnehmer* ist FS und bezieht sich auf den PS von *Versicherungsnehmer*.
- *Fahrzeug* ist FS und bezieht sich auf den PS von *Fahrzeug*.
- *Mitarbeiter* ist FS und bezieht sich auf den PS von *Mitarbeiter*.
- Attribut *Art* kann die Werte *HP* (Haftpflicht), *TK* (Teilkasko) und *VK* (Vollkasko) annehmen.

- *Mitarbeiter* ist FS und bezieht sich auf den PS von *Mitarbeiter*.
- *Schadensfall* ist FS und bezieht sich auf den PS von *Schadensfall*.
- *Fahrzeug* ist FS und bezieht sich auf den PS von *Fahrzeug*.

(a) Herr Meier schließt eine Teilkaskoversicherung bei „Insurance Pro“ am 11.11.2011 für seinen neuen Wagen, einen roten VW Golf VII mit amtlichem Kennzeichen BT-BT 2011, ab. Der Vertrag erhält die laufende Nummer 1631 und wird von Frau Schmied mit der Personalnummer 27 bearbeitet. Herr Meier erhält die Kundennummer 588, da er bisher noch kein Kunde war.

Geben Sie die SQL-Befehle an, um die neue Versicherung in die Datenbank von „Insurance Pro“ einzutragen. Werte, die hierbei nicht bekannt sind, sollen weggelassen werden, wobei angenommen werden darf, dass die entsprechenden Attribute nicht mit der Bedingung NOT NULL versehen sind.

```sql
INSERT INTO Fahrzeug
  (Kennzeichen, Farbe, Fahrzeugtyp) VALUES
  ('BT-BT 2011', 'rot', 'VW Golf VII');

INSERT INTO Versicherungsnehmer
  (KNr, Name)  VALUES
  (588, 'Meier');

INSERT INTO Versicherungsvertrag
  (VNr, Abschlussdatum, Art, Versicherungsnehmer, Fahrzeug, Mitarbeiter) VALUES
  (1631, 11.11.2011, 'TK', 588, 'BT-BT 2011', 27);
```

(b) Beschreiben Sie umgangssprachlich, wonach mit folgendem Ausdruck gesucht wird:

$\pi_{\text{Versicherungsnehmer}}(\sigma_{\text{Fahrzeugtyp}=\text{'Fiat500'}}(\text{FZ} \bowtie_{\rho_{\text{Kennzeichen}\leftarrow\text{Fahrzeug}}} (\text{VV})))$

Anmerkung: Die Abkürzung *FZ* steht für die Tabelle *Fahrzeug* und *VV* für die Tabelle *Versicherungsvertrag*.

> Es wird nach der Kundennummer aller Versicherungsnehmer gesucht, die einen Versicherungsvertrag für einen „Fiat 500“ abgeschlossen haben.

(c) Die Angaben der Mitarbeiter (Nr., Name und Kontakt), deren Abteilung ihren Sitz in München oder Stuttgart hat, sollen explizit alphabetisch nach Namen sortiert ausgegeben werden. Geben Sie einen SQL-Befehl hierfür an.

```sql
SELECT m.MNr AS Nr, m.Name, m.Kontaktdaten AS Kontakt
FROM Mitarbeiter m, Abteilung a
WHERE
  a.ANr = m.Abteilung AND
  a.Ort IN ('München', 'Stuttgart')
ORDER BY m.Name;
```

(d) Es soll ausgegeben werden, wie oft jeder Versicherungsnehmer (KNr, Name, Anschrift) an einem Schadensfall beteiligt war. Hierbei sollen nur die Kunden, die an mindestens drei Schadensfällen beteiligt waren, in absteigender Reihenfolge aufgelistet werden.

```sql
SELECT vn.KNr, vn.Name, vn.Anschrift, COUNT(*) AS Schadensfaelle
FROM
  Versicherungsnehmer vn,
  Schadensfall s,
  Beteilung b,
  Versicherungsvertrag vv
WHERE
  vn.KNr = vv.Versicherungsnehmer AND
  s.SNr = b.Schadensfall AND
  b.Fahrzeug = vv.Fahrzeug
GROUP BY vn.KNr, vn.Name, vn.Anschrift
HAVING COUNT(*) >= 3
ORDER BY Schadensfaelle DESC;
```

Es ist nicht genau angegeben, nach was sortiert werden soll. Ich sortiere nach Anzahl an Schadensfällen, weil dass meiner Meinung nach am meisten Sinn macht.

Thema Nr. 2

Teilaufgabe Nr. 1

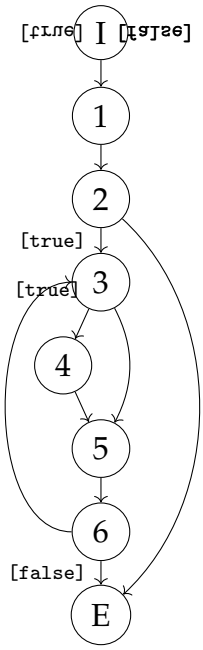
Aufgabe 2 [Methode „isPalindrom()“]

Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:
Abkürzungen: I = Import, E = Export

```java
boolean isPalindrom(String s) {
  boolean yesItIs = true;
  if (s != null && s.length() > 1) {
    do {
      if (s.charAt(0) != s.charAt(s.length() -
      ↪  1)) {
        yesItIs = false;
      }
      s = s.substring(1, s.length() - 1);

    } while (yesItIs && s.length() > 1);
  }
  return yesItIs;
}
```

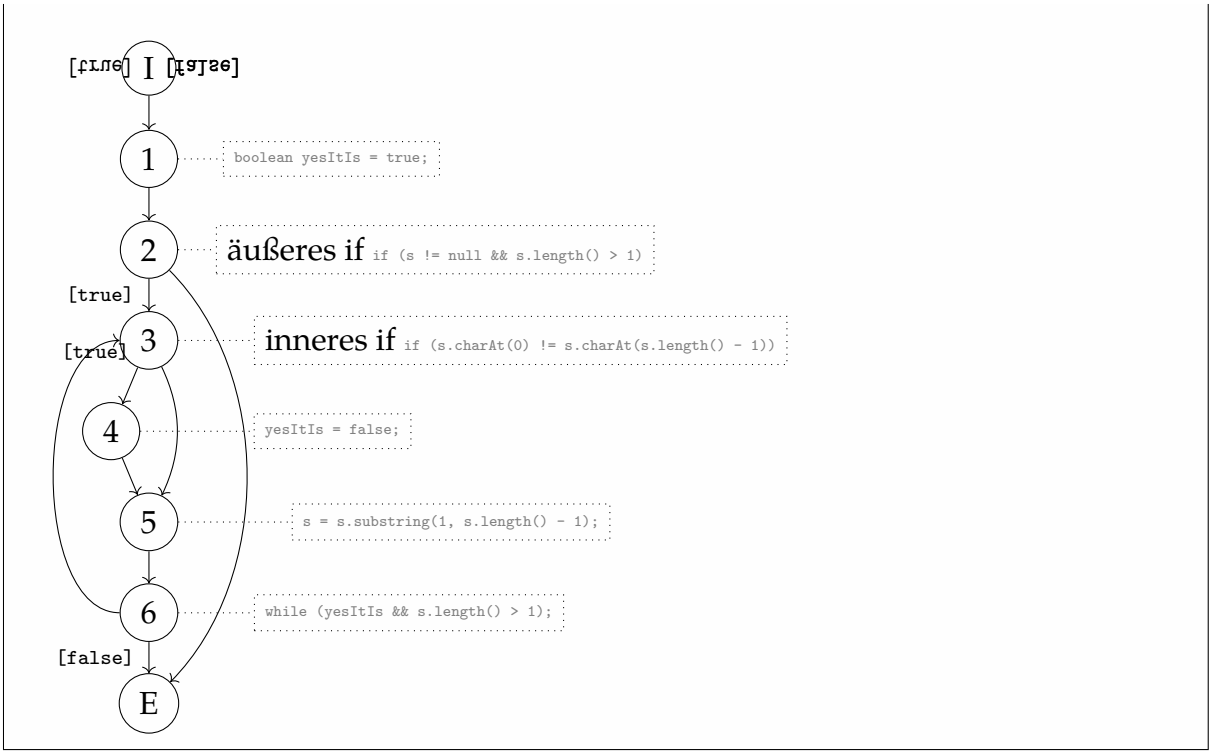
Code-Beispiel auf Github ansehen:
`src/main/java/org/bschlangaul/aufgaben/sosy/pu_5/Aufgabe2.java`



(a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen ... mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

Lösungsvorschlag

> Bemerkung: In der Aufgabenstellung steht *„Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, [...] “*. Das bedeutet, dass es hier erstmal egal ist, ob ein Pfad im Code möglich ist oder nicht!



(i) Verzweigungsüberdeckung (Branch-Coverage, C_1)

Lösungsvorschlag

Pfad 1 (p1) (I) - (1) - (2) - (E)
(äußere `if`-Bedingung `false`)

Pfad 2 (p2) (I) - (1) - (2) - (3) - (5) - (6) - (3) - (4) - (5) - (6) - (E)
(äußere `if`-Bedingung `true`, innere `if`-Bedingung `false`, Wiederholung, innere `if`-Bedingung `true`, keine Wiederholung)

(ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage, $C_{\infty,2}$)

ohne Ausführung der Wiederholung (äußere Pfade): p1 (siehe oben) Ⓘ - ① - ② - Ⓔ

Boundary-Test: (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

interior-Test: (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

innere `if`-Bedingung `true`: ③ - ④ - ⑤ - ⑥

innere `if`-Bedingung `false`: ③ - ⑤ - ⑥

p5 Ⓘ - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `true`, innere `if`-Bedingung `true`)

p2 (siehe oben) Ⓘ - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `false`, innere `if`-Bedingung `true`)

p6 Ⓘ - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `true`, innere `if`-Bedingung `false`)

p7 ⓘ - ① - ② - ③ - ⑤ - ⑥ - ③ - ⑤ - ⑥ - Ⓔ
(innere `if`-Bedingung `false`, innere `if`-Bedingung `false`)

mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

(b) Welche der vorangehend ermittelten Pfade für die $C_{\infty,2}$-Überdeckung sind mittels Testfällen tatsächlich überdeckbar (*„feasible“*)? Falls der Pfad ausführbar ist, geben Sie den zugehörigen Testfall an - andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

p1 `s = "a";`

p2 `s = "abaa";`

p3 `s = "ab";`

p4 `s = "aa";`

p5 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`)
keine Wiederholung!

p6 nicht überdeckbar, da `yesItIs = false`, wenn innere `if`-Bedingung `true`)
keine Wiederholung!

p7 `s = "abba";`

(c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

$M = b + p = 3 + 1 = 4$

(b: Anzahl Binärverzweigungen, p: Anzahl Zusammenhangskomponenten)

Alternativ

$M = e - n + 2p = 10 - 8 + 2 = 4$

(e: Anzahl Kanten, n: Anzahl Knoten, p: Anzahl Zusammenhangskomponenten)

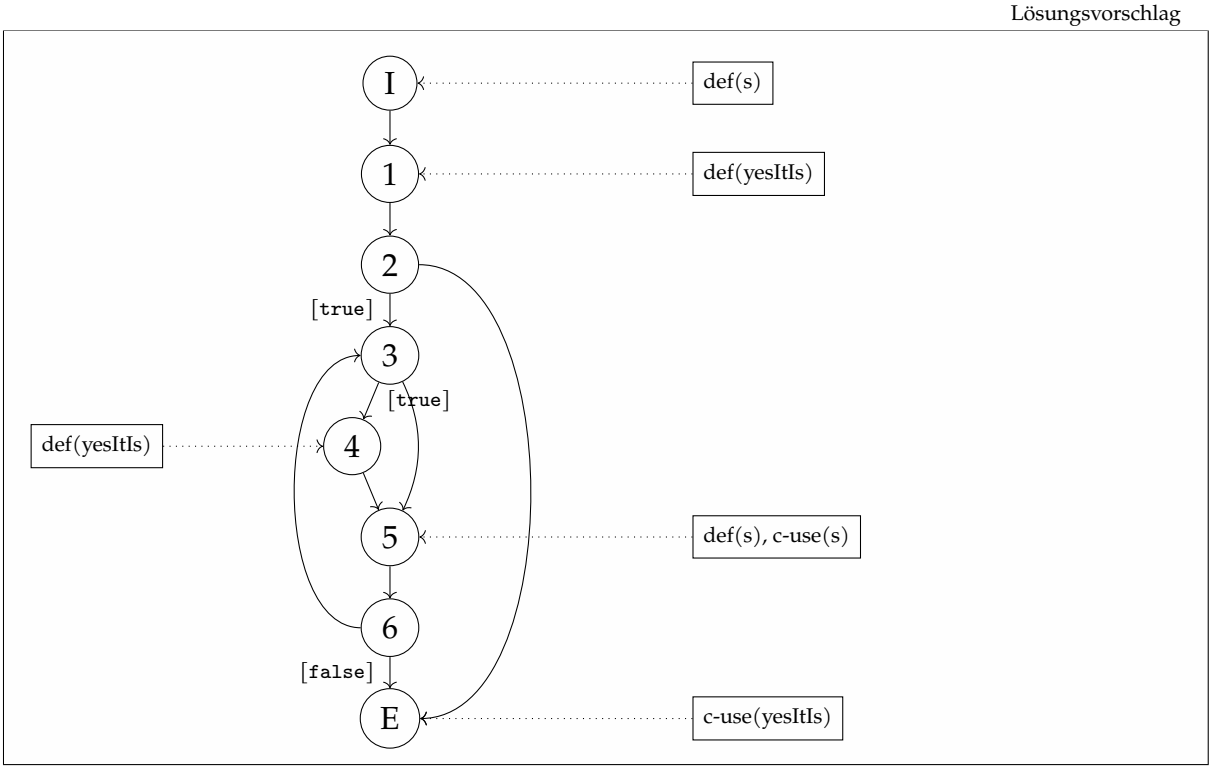
(d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Lösungsvorschlag

Eine 100%-ige Pfadüberdeckung kann nicht erzielt werden, da es zum einen unüberdeckbare Pfade gibt (vgl. Teilaufgabe b). Zum anderen ist das Testen aller Testfälle nicht möglich, da die Anzahl an Zeichen des übergebenen Wortes nicht begrenzt ist und es somit eine unendliche Anzahl an Testfällen gibt.

(e) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (`c-use`) oder prädikativ (`p-use`) sind.

Lösungsvorschlag



Aufgabe 3 [ASCII]

Sei wp(A, Q) die schwächste Vorbedingung (weakest precondition) eines Programmfragments A bei gegebener Nachbedingung Q so, dass A alle Eingaben, die wp(A,Q) erfüllen, auf gültige Ausgaben abbildet, die Q erfüllen.

Bestimmen Sie schrittweise und formal (mittels Floyd-Hoare-Kalkül) jeweils wp(A, Q) für folgende Code-Fragmente A und Nachbedingungen Q und vereinfachen Sie dabei den jeweils ermittelten Ausdruck so weit wie möglich.

Die Variablen x, y und z in folgenden Pseudo-Codes seien ganzzahlig (vom Typ int). Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

(a) Sequenz:

```
x = -2 * (x + 2 * y);
y += 2 * x + y + z;
z -= x - y - z;
```

$Q :\equiv x = y + z$

Lösungsvorschlag

Code umformulieren:

```
x = -2 * (x + 2 * y);
y = y + 2 * x + y + z;
z = z - (x - y - z);
```

wp(`"x=-2*(x+2*y);y=2*y+2*x+z;z=z-(x-y-z);"`, $x = y + z$)

z eingesetzen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = y + (z - (x - y - z)))$$

Innere Klammer auflösen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = y + (-x + y - 2z))$$

Klammer auflösen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, x = -x + 2y + 2z)$$

$-x$ auf beiden Seiten

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, 0 = -2x + 2y + 2z)$$

$\div 2$ auf beiden Seiten

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);y=2*y+2*x+z;"}, 0 = -x + y + z)$$

y einsetzen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);"}, 0 = -x + (2y + 2x + z) + z)$$

Term vereinfachen

$$\equiv \quad \mathrm{wp}(\texttt{"x=-2*(x+2*y);"}, 0 = x + 2y + 2z)$$

x einsetzen

$$\equiv \quad \mathrm{wp}(\texttt{""}, 0 = (-2(x + 2y)) + 2y + 2z)$$

wp weglassen

$$\equiv \quad 0 = (-2(x + 2y)) + 2y + 2z$$

ausmultiplizieren

$$\equiv \quad 0 = (-2x - 4y) + 2y + 2z$$

Klammer auflösen, vereinfachen

$$\equiv \quad 0 = -2x - 2y + 2z$$

$\div 2$ auf beiden Seiten

$$\equiv \quad 0 = -x - y + z$$

x nach links holen mit $+x$ auf beiden Seiten

$$\equiv \quad x = -y + z$$

y ganz nach links schreiben

$$\equiv \quad x = z - y$$

$$x = -2 \cdot (x + 2 \cdot y)$$

(b) Verzweigung:

```
if (x < y) {
  x = y + z;
} else if (y > 0) {
  z = y - 1;
} else {
  x -= y -= z;
}
```

$Q :\equiv x > z$

Lösungsvorschlag

1. Fall: $x < y$

2. Fall: $x \geq y \land y > 0$

3. Fall: $x \geq y \land y \leq 0$

Code umformulieren:

```
if (x < y) {
  x = y + z;
} else if (x >= y && y > 0) {
  z = y - 1;
} else {
  y = y - z;
  x = x - y;
}
```

$\mathrm{wp}(\texttt{"if(x<y){x=y+z;}else if(x>=y\&\&y>0){z=y-1;}else{y=y-z;x=x-y;}"}, x > z)$

$\equiv$ (In mehrere kleinere wp-Kalküle aufsplitten)

$$\Big((x < y) \land \mathrm{wp}(\texttt{"x=y+z;"}, x > z)\Big) \lor$$
$$\Big((x \geq y \land y > 0) \land \mathrm{wp}(\texttt{"z=y-1;"}, x > z)\Big) \lor$$
$$\Big((x \geq y \land y \leq 0) \land \mathrm{wp}(\texttt{"y=y-z;x=x-y;"}, x > z)\Big)$$

$\equiv$ (Code einsetzen)

$$\Big((x < y) \land \mathrm{wp}(\texttt{""}, y + z > z)\Big) \lor$$
$$\Big((x \geq y \land y > 0) \land \mathrm{wp}(\texttt{""}, x > y - 1)\Big) \lor$$
$$\Big((x \geq y \land y \leq 0) \land \mathrm{wp}(\texttt{"y=y-z;"}, x - y > z)\Big)$$

$\equiv$ (wp-Kalkül-Schreibweise weg lassen, Code weiter einsetzen)

$$\Big((x < y) \land y + z > z\Big) \lor$$
$$\Big((x \geq y \land y > 0) \land x > y - 1\Big) \lor$$
$$\Big((x \geq y \land y \leq 0) \land \mathrm{wp}(\texttt{""}, x - (y - z) > z)\Big)$$

$\equiv$ (Terme vereinfachen, wp-Kalkül-Schreibweise weg lassen)

$$\Big(x < y \land y > 0\Big) \lor$$
$$\Big(x \geq y^a \land y > 0\Big) \lor$$
$$\Big((x \geq y \land y \leq 0) \land x - (y - z) > z\Big)$$

$\equiv$ (letzten Term vereinfachen)

$$\Big(x < y \land y > 0\Big) \lor$$
$$\Big(x \geq y \land y > 0\Big) \lor$$
$$\Big((x \geq y \land y \leq 0) \land x - y > 0\Big)$$

$\equiv$ (ein $\land$ eleminieren)

$$\Big(x < y \land y > 0\Big) \lor$$
$$\Big(x \geq y \land y > 0\Big) \lor$$
$$\Big(y \leq 0 \land x > y\Big)$$

[a] $x > y - 1 \land x \geq y$ ergibt $x \geq y$
[a] $x > y - 1 \land x \geq y$ ergibt $x \geq y$

(c) Mehrfachauswahl:

```
switch (z) {
  case "x":
    y = "x";
  case "y":
    y = --z;
    break;
  default:
    y = 0x39 + "?";
}
```

$Q :\equiv$ `'x'` $= y$

Hinweis zu den ASCII-Codes

- `'x'` $= 120_{(10)}$
- `'y'` $= 121_{(10)}$
- `0x39` $= 57_{(10)}$
- `'?'` $= 63_{(10)}$

Lösungsvorschlag

Mehrfachauswahl in Bedingte Anweisungen umschreiben. Dabei beachten, dass bei fehlendem `break` die Anweisungen im folgenden Fall bzw. ggf. in den folgenden Fällen ausgeführt werden:

```
if (z == "x") {
  y = "x";
  y = z - 1;
} else if (z == "y") {
  y = z - 1;
} else {
  y = 0x39 + "?";
}
```

Da kein `break` im Fall `z == "x"`. `--z` bedeutet, dass die Variable erst um eins verringert und dann zugewiesen wird.

```
if (z == 120) {
  y = 120;
  y = 120 - 1;
} else if (z == 121) {
  y = 121 - 1;
} else {
  y = 57 + 63;
}
```

Vereinfachung / Zusammenfassung:

```
if (z == 120) {
  y = 120;
  y = 119;
} else if (z == 121) {
  y = 120;
} else {
  y = 120;
}
```

$$\text{wp}(\texttt{"if(z==120)\{y=120;y=119;\}else if(z==121)\{y=120;\}else\{y=120;\}"}, 120 = y)$$

$\equiv$ (In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned}&\Big((z = 120) \land \text{wp}(\texttt{"y=120;y=119;"}, 120 = y)\Big) \lor \\ &\Big(((z \neq 120) \land (z = 121)) \land \text{wp}(\texttt{"y=120;"}, 120 = y)\Big) \lor \\ &\Big(((z \neq 120) \land (z \neq 121)) \land \text{wp}(\texttt{"y=120;"}, 120 = y)\Big)\end{aligned}$$

$\equiv$ (Code einsetzen)

$$\begin{aligned}&\Big((z = 120) \land \text{wp}(\texttt{"y=120;"}, 120 = 119)\Big) \lor \\ &\Big(((z \neq 120) \land (z = 121)) \land \text{wp}(\texttt{""}, 120 = 120)\Big) \lor \\ &\Big(((z \neq 120) \land (z \neq 121)) \land \text{wp}(\texttt{""}, 120 = 120)\Big)\end{aligned}$$

$\equiv$ (vereinfachen)

$$\begin{aligned} &\text{false} \vee \\ &\Big((z = 121) \wedge \text{true}\Big) \vee \\ &\Big(((z \neq 120) \wedge (z \neq 121)) \wedge \text{true}\Big) \end{aligned}$$

$\equiv \text{false} \vee (z = 121) \vee ((z \neq 120) \wedge (z \neq 121))$
$\equiv (z = 121) \vee (z \neq 121)$
$\equiv z \neq 121$
Alle Zahlen außer 120 sind möglich bzw. alle Zeichen außer 'x'.