Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

27. Juli 1995

Aufgaben zur Klausur **Grundlagen der Programmierung**, **Software Engineering** und **Logische Programmierung** im SS 95 (WI03)

Zeit: 120 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 7 Seiten

---

Aufgabe 1:

Gegeben sei das folgende Programm zum Suchen von ganzzahligen Lösungen für eine Gleichung der Form $c_1 * x + c_2 = c_3 * y$ c_1, c_2 und c_3 seien dabei ganzzahlige Koefizienten > 0.

$suche(x : \mathsf{N}_1; y : \mathsf{N}_1) : \mathsf{N}_1$
    if $c_1 * x + c_2 = c_3 * y$
    then x
    else
        if $c_1 * x + c_2 > c_3 * y$
        then $suche(x, y + 1)$
        else $suche(x + 1, y)$

Terminiert dieser Algorithmus immer?

ja ☐ nein ☐

Begründung:

.....

.....

Entwickeln Sie zu diesem Programm durch systematische Transformation gemäß der Transformationsschemata aus der Vorlesung ein gleichwertiges Programm, das mit einer while Schleife arbeitet.

.....

.....

.....

.....

.....

.....

.....

.....

In der entstandenen Schleife werden Operationen (unnötigerweise) mehrfach ausgewertet, insbesondere sind in der Schleife, in der Schleifenbedingung und im Rumpf, Multplikationen enthalten. Verfeinern Sie den entstandenen Algorithmus so, daß keine Multiplikationen in der Schleife benötigt werden, die Struktur des Programms soll aber erhalen bleiben.

Aufgabe 2:

Gegeben seien die Felder f und g

var

f : array $[0..n-1]$ of N_0
g : array $[0..n-1]$ of N_0

mit $n > 0$ und die folgenden prädikatenlogischen Formeln

1. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = g[j]$
2. $\forall\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] = g[j]$
3. $\exists\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = g[j]$
4. $\exists\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] = g[j]$
5. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] \neq g[j]$
6. $\forall\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] \neq g[j]$
7. $\exists\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] \neq g[j]$
8. $\exists\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] \neq g[j]$
9. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] \neq f[j]$
10. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = f[j] \wedge g[i] = g[j]$
11. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = g[i] \wedge f[j] = g[j]$

Hinweis zur Notation:

$\forall\, 0 \leq i < n \bullet P(i)$

ist gleichwertig mit

$(\forall\, 0 \leq i < n)(P(i))$

analoges gilt für $\exists$.

Geben sie für die folgenden Aussagen die Nummer(n) von gleichwertigen Formeln an, Mehrfachnennungen sind möglich, gibt es keine Formel tragen Sie 0 an die vorgesehene Stelle ein.

1. Alle Elemente in f kommen auch in g vor.

   .....

2. In f sind die gleichen Elemente wie in g gespeichert.

   .....

3. Alle Elemente in g sind gleich und dieser Wert kommt auch in f vor.

   .....

4. Alle Elemente in g sind gleich.

   .....

5. f enthält an allen Positionen den gleichen Wert wie g an der entsprechenden Position.

   .....

6. Kein Element aus f kommt in g vor und umgekehrt.

   .....

7. Mindestens an einer Position ist g von f verschieden.

   .....

8. Ein Element aus f kommt nicht in g vor.

   .....

9. f enthält an jeder Position den gleichen Wert und g enthält an jeder Position den gleichen Wert.

   .....

10. false

    .....

Aufgabe 3:

Gegeben sei folgendes Programm zur Berechnung von Wegen in einem Graphen, die Kanten in einem Graphen werden durch kante/2 mit Anfangs- und Endknoten als 1. und 2. Argument repräsentiert.

Das Prädikat weg(+AnfangsKnoten, +EndKnoten, -Weg) berechnet Wege in einem Graphen. Es deligiert seine Arbeit an weg1/3.

```
kante(a, b).
kante(b, a).
kante(b, c).
kante(c, d).

weg(X, Y, W) :-
        weg1(X, Y, W).

weg1(X, Y, [X, Y]) :-
        kante(X, Y).

weg1(X, Y, [X | W1]) :-
        kante(X, Z),
        weg1(Z, Y, W1).
```

1. Wieviele Lösungen bekommt man für die Anfrage ?- weg(c,d,W)

   .....

   und wie sieht die 1. Lösung aus

   .....

2. Wieviele Lösungen bekommt man für die Anfrage ?- weg(a,c,W)

   .....

   und wie sieht die 2. Lösung aus

   .....

3. Wieviele Lösungen bekommt man für die Anfrage ?- weg(a,d,W)

   .....

   und wie sieht die 1. Lösung aus

   .....

Erweitern sie das Prädikat `weg1/3` zu einem Prädikat `weg1/4`, das nur noch zyklenfreie Weg in einem Graphen berechnet

.....

.....

.....

.....

.....

.....

.....

.....

Modifizieren Sie die Regel für `weg/3` so, daß sie mit `weg1/4` arbeitet

.....

.....

Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

27. Juli 1995

Aufgaben zur Klausur **Grundlagen der Programmierung** und **Software Engineering** im SS 95 (II13)

Zeit: 120 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 6 Seiten

---

Aufgabe 1:

Gegeben sei das folgende Programm zum Suchen von ganzzahligen Lösungen für eine Gleichung der Form $c_1 * x + c_2 = c_3 * y$ c_1, c_2 und c_3 seien dabei ganzzahlige Koefizienten > 0.

$suche(x : \mathsf{N}_1; y : \mathsf{N}_1) : \mathsf{N}_1$
    if $c_1 * x + c_2 = c_3 * y$
    then x
    else
        if $c_1 * x + c_2 > c_3 * y$
        then $suche(x, y + 1)$
        else $suche(x + 1, y)$

Terminiert dieser Algorithmus immer?

ja ☐ nein ☐

Begründung:

.....

.....

Entwickeln Sie zu diesem Programm durch systematische Transformation gemäß der Transformationsschemata aus der Vorlesung ein gleichwertiges Programm, das mit einer while Schleife arbeitet.

.....

.....

.....

.....

.....

.....

.....

.....

In der entstandenen Schleife werden Operationen (unnötigerweise) mehrfach ausgewertet, insbesondere sind in der Schleife, in der Schleifenbedingung und im Rumpf, Multplikationen enthalten. Verfeinern Sie den entstandenen Algorithmus so, daß keine Multiplikationen in der Schleife benötigt werden, die Struktur des Programms soll aber erhalen bleiben.

Aufgabe 2:

Gegeben seien die Felder f und g

var

f : array $[0..n-1]$ of N_0
g : array $[0..n-1]$ of N_0

mit $n > 0$ und die folgenden prädikatenlogischen Formeln

1. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = g[j]$
2. $\forall\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] = g[j]$
3. $\exists\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = g[j]$
4. $\exists\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] = g[j]$
5. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] \neq g[j]$
6. $\forall\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] \neq g[j]$
7. $\exists\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] \neq g[j]$
8. $\exists\, 0 \leq i < n \bullet \exists\, 0 \leq j < n \bullet f[i] \neq g[j]$
9. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] \neq f[j]$
10. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = f[j] \wedge g[i] = g[j]$
11. $\forall\, 0 \leq i < n \bullet \forall\, 0 \leq j < n \bullet f[i] = g[i] \wedge f[j] = g[j]$

Hinweis zur Notation:

$\forall\, 0 \leq i < n \bullet P(i)$

ist gleichwertig mit

$(\forall\, 0 \leq i < n)(P(i))$

analoges gilt für $\exists$.

Geben sie für die folgenden Aussagen die Nummer(n) von gleichwertigen Formeln an, Mehrfachnennungen sind möglich, gibt es keine Formel tragen Sie 0 an die vorgesehene Stelle ein.

1. Alle Elemente in f kommen auch in g vor.

   .....

2. In f sind die gleichen Elemente wie in g gespeichert.

   .....

3. Alle Elemente in g sind gleich und dieser Wert kommt auch in f vor.

   .....

4. Alle Elemente in g sind gleich.

   .....

5. f enthält an allen Positionen den gleichen Wert wie g an der entsprechenden Position.

   .....

6. Kein Element aus f kommt in g vor und umgekehrt.

   .....

7. Mindestens an einer Position ist g von f verschieden.

   .....

8. Ein Element aus f kommt nicht in g vor.

   .....

9. f enthält an jeder Position den gleichen Wert und g enthält an jeder Position den gleichen Wert.

   .....

10. false

    .....

Aufgabe 3:

Zeigen Sie, daß das folgende Programmstück korrekt ist bezüglich der angegebenen Vor- und Nachbedingung, d.h. daß es die Werte in a und b vertauscht und negiert. Verwenden Sie hierfür die Beweisregeln für Zuweisungen und Anweisungsfolgen und die elementare arithmetische Umformungen.

Das Programm:

$$
\begin{array}{ll}
 & \{\ V\ \} \\
 & \{\ P_0\ \} \\
a := a - b; & \{\ P_{1a}\ \} \\
 & \{\ P_1\ \} \\
b := b - a; & \{\ P_2\ \} \\
a := a - b & \{\ P\ \}
\end{array}
$$

Die Vorbedingung V: $(a = v_1) \wedge (b = v_2)$

Die Nachbedingung P: $(a = -v_2) \wedge (b = -v_1)$

Die Bedingung P_2:

.....

Die Bedingung P_1:

.....

Die vereinfachte Bedingung P_{1a}:

.....

Die Bedingung P_0:

.....

Die vereinfachte gleichwertige Bedingung V:

.....

Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

27. Juli 1995

Aufgaben zur Klausur **Logische Programmierung** im SS 95 (IA41)

Zeit: 60 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 3 Seiten

---

Aufgabe 1:

Gegeben sei folgendes Programm zur Berechnung von Wegen in einem Graphen, die Kanten in einem Graphen werden durch kante/2 mit Anfangs- und Endknoten als 1. und 2. Argument repräsentiert.

Das Prädikat weg(+AnfangsKnoten, +EndKnoten, -Weg) berechnet Wege in einem Graphen. Es deligiert seine Arbeit an weg1/3.

```
kante(a, b).
kante(b, a).
kante(b, c).
kante(c, d).

weg(X, Y, W) :-
        weg1(X, Y, W).

weg1(X, Y, [X, Y]) :-
        kante(X, Y).

weg1(X, Y, [X | W1]) :-
        kante(X, Z),
        weg1(Z, Y, W1).
```

1. Wieviele Lösungen bekommt man für die Anfrage ?- weg(c,d,W)

   .....

   und wie sieht die 1. Lösung aus

   .....

2. Wieviele Lösungen bekommt man für die Anfrage ?- weg(a,c,W)

   .....

   und wie sieht die 2. Lösung aus

   .....

3. Wieviele Lösungen bekommt man für die Anfrage ?- weg(a,d,W)

   .....

   und wie sieht die 1. Lösung aus

   .....

Erweitern sie das Prädikat `weg1/3` zu einem Prädikat `weg1/4`, das nur noch zyklenfreie Weg in einem Graphen berechnet

.....

.....

.....

.....

.....

.....

.....

.....

Modifizieren Sie die Regel für `weg/3` so, daß sie mit `weg1/4` arbeitet

.....

.....

Fachhochschule Wedel
Prof. Dr. Uwe Schmidt

15. September 1995

Aufgaben zur Nachklausur **Grundlagen der Programmierung**, **Software Engineering** und **Logische Programmierung** im SS 95 (WI03)

Zeit: 120 Minuten erlaubte Hilfsmittel: keine

Bitte tragen Sie Ihre Antworten und fertigen Lösungen ausschließlich an den gekennzeichneten Stellen in das Aufgabenblatt ein. Ist ihre Lösung wesentlich umfangreicher, so überprüfen Sie bitte nochmals Ihren Lösungsweg.

Viel Erfolg !

Diese Klausur besteht einschließlich dieses Deckblattes aus 6 Seiten

---

Aufgabe 1:

Bei der Berechnung eines Polynoms erfolgt eine (komplizierte) Form der Aufsummierung von Elementen eines Feldes.

Seien folgende Variablen gegeben:

var
a : array $[0..n]$ of R;
x : R;
y : R;
i : $\mathbb{N}_0$

In a seien die Koefizienten für ein Polynom vom Grad n gegeben und in x die Stelle, an der das Polynom berechnet werden soll.

Schreiben Sie ein Programmstück zur Berechnung des Polynoms a an der Stelle x, das nach dem folgenden Schema arbeitet

$y = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$

Das Programmstück soll mit einer while Schleife arbeiten, die zu berechneden Potenzen sollen mit Hilfe einer Funktion *power* berechnet werden

Dieser Algorithmus erfordert die Berechnung von Potenzen. Entwickeln Sie für die Berechnung der Potenzen die Funktion *power*, diese soll auch wieder mit einer while Schleife arbeiten.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Welche Vorteile hat die Zerlegung in Teilalgorithmen gegenüber dem direkten Einsetzen.

.....

.....

.....

Schreiben Sie ein zweites Programmstück zur Berechnung eines Polynoms, das nach dem Hornerschema arbeitet (abwechselnde Multiplikation und Addition)

$y = ((\ldots(a_n x + a_{n-1})x + \ldots)x + a_1)x + a_0$

Warum ist dieser Algorithmus schneller als der erste?

Aufgabe 2:

Zeigen Sie, daß das folgende Programmstück korrekt ist bezüglich der angegebenen Vor- und Nachbedingung, d.h. daß es die Werte in a und b vertauscht und negiert. Verwenden Sie hierfür die Beweisregeln für Zuweisungen und Anweisungsfolgen und die elementare arithmetische Umformungen.

Das Programm:

$$
\begin{array}{ll}
 & \{\ V\ \} \\
 & \{\ P_0\ \} \\
a := a - b; & \{\ P_{1a}\ \} \\
 & \{\ P_1\ \} \\
b := b - a; & \{\ P_2\ \} \\
a := a - b & \{\ P\ \}
\end{array}
$$

Die Vorbedingung V: $(a = v_1) \wedge (b = v_2)$

Die Nachbedingung P: $(a = -v_2) \wedge (b = -v_1)$

Die Bedingung P_2:

.....

Die Bedingung P_1:

.....

Die vereinfachte Bedingung P_{1a}:

.....

Die Bedingung P_0:

.....

Die vereinfachte gleichwertige Bedingung V:

.....

Aufgabe 3:

Schreiben Sie ein 2-stelliges Prädikat $rmduplicates(L, R)$, das aus der Liste L eine neue Liste R aufbaut, in der alle Elemente von L enthalten sind, aber keine Duplikate vorkommen.

Benutzen Sie hierzu ein Prädikat $insert(E, S, S1)$, das das Element E zu der Liste S hinzufügt und in $S1$ zurückgibt, falls es noch nicht enthalten ist. Dieses Prädikat soll deterministisch arbeiten. Das *member*–Prädikat darf verwendet werden. Verwenden Sie, falls notwendig, keinen Cut sondern das *not* Prädikat.

$rmduplicates(+L, -R)$:

.....

.....

.....

.....

.....

$insert(+E, +S, -S1)$:

.....

.....

.....

.....

.....

Hat die Verwendung des *not* Prädikats gegenüber dem Cut Vorteile?

ja ☐ nein ☐

Begründung:

.....

.....