Name: | Vorname: | Matr.–Nr.:

Technische Universität München
Fakultät für Informatik
Prof. Dr. H. Seidl

SoS 2005
12. Oktober 2005

Wiederholungsklausur zu Einführung in die Informatik II

Hinweis: In dieser Klausur können Sie insgesamt 60 Punkte erreichen. Zum Bestehen benötigen Sie mindestens 24 Punkte.

Aufgabe 1 Java-GUI (10 Punkte)

Programmieren Sie eine **Java-Applikation**, die ein Fenster anzeigt auf dem sich ein Text-Eingabe-Feld und ein Knopf mit der Aufschrift „Quadriere“befindet. Wird der Knopf betätigt, so soll die Eingabe (sofern es sich um eine Zahl handelt) quadriert werden und das Ergebnis der Quadratur im Text-Eingabe-Feld erscheinen.

Hinweis:

Zum Setzen bzw. zum Zurückliefern des aktuellen Inhaltes stellt die Klasse `TextField` die Methode `String getText()` bzw. `void setText(String s)` bereit.

Zur Konvertierung eines Strings in einen Integer-Wert stellt die Klasse `Integer` die statische Methode `int parseInt(String s)` bereit. Diese wirft eine Exception, falls der String `s` keinen Integer-Wert repräsentiert.

Aufgabe 2 Datalog **(1+1+3+2+3=10 Punkte)**

Die Prädikate `kind`, `maennlich`, `weiblich` seien durch folgende Fakten definiert.

```
maennlich(pepone).          maennlich(rodrigo).
maennlich(wilpert).         maennlich(hieronymus).
maennlich(etienne).

weiblich(konstanze).        weiblich(rosalinde).
weiblich(valeska).          weiblich(harriet).
weiblich(bernadette).       weiblich(margarete).
weiblich(evi).

kind(rodrigo,pepone).       kind(rodrigo,konstanze).
kind(rosalinde,pepone).     kind(rosalinde,konstanze).

kind(evi,rodrigo).          kind(evi,margarete).

kind(hieronymus,rosalinde). kind(hieronymus,wilpert).

kind(etienne,hieronymus).   kind(etienne,valeska).

kind(bernadette,etienne).   kind(bernadette,harriet).
```

Ein Faktum `kind(a,b)` drückt hierbei die Tatsache aus, dass `a` ein Kind von `b` ist.

a) Definieren Sie ein 2-stelliges Prädikat `sohn`, das von einem Tupel (a,b) genau dann erfüllt wird, wenn *a* ein Sohn von *b* ist.

b) Definieren Sie ein 2-stelliges Prädikat `mutter`, das von einem Tupel (a,b) genau dann erfüllt wird, wenn *a* die Mutter von *b* ist.

c) Definieren Sie ein 2-stelliges Prädikat `nachfahre`, das von einem Tupel (a,b) genau dann erfüllt wird, wenn *a* ein (direkter oder indirekter) Nachfahre von *b* ist.

d) Definieren Sie ein 2-stelliges Prädikat `gemeinsames_kind`, das von einem Tupel (a,b) genau dann erfüllt wird, wenn *a* und *b* ein gemeinsames Kind haben.

e) Geben Sie eine Ableitung für das Faktum `nachfahre(hieronymus,konstanze)` bezüglich der Regeln aus Teilaufgabe c) an.

Aufgabe 3 Ocaml **(3+3+4=10 Punkte)**

a) Definieren Sie eine Ocaml-Funktion

```
map2 : 'a list -> 'b list -> ('a -> 'b -> 'c) -> 'c list
```

die als Argumente zwei gleichlange Listen `l1` und `l2` und eine Funktion `f` erhält. Die Funktion `f` erwartet zwei Argumente, eins vom Typ der Elemente der Liste `l1`, das andere vom Typ der Elemente der Liste `l2` und liefert einen beliebigen Wert zurück. Der Aufruf `map2 l1 l2 f` soll die Liste zurückliefern, die an der Stelle *i* den Wert hat, den `f` angewendet auf die `i`-ten Elemente aus `l1` und `l2` berechnet.

Zum Beispiel soll `map2 [1;2;3] [50;60;70] (fun x y -> x+y)` die Liste `[51;62;73]` zurückliefern.

b) Definieren Sie eine Funktion `zip : 'a list -> 'b list -> ('a * 'b) list`, die zwei gleichlange Listen `l1` und `l2` als Argumente erhält und die Liste zurückliefert, die an der Stelle *i* ein Paar bestehend aus den *i*-ten Elementen der Listen `l1` und `l2` hat.

Zum Beispiel soll `zip [1;2;3] [50;60;70]` die Liste `[(1,50);(2,60);(3,70)]` zurückliefern.

c) Gegeben sei der wie folgt definierte Daten-Typ `graph` zur Repräsentation gerichteter Graphen:

```
type node = int
type graph = node list array
```

Bemerkung: Der hier angegebene Daten-Typ entspricht dem in der Vorlesung angegebenen Daten- Typ `'a graph`, bis auf die Tatsache, dass die Kanten hier nicht mit einer Information versehen werden können.

Definieren Sie eine Funktion `inverse : graph -> graph`, die einen Graphen g als Argument erhält und einen Graphen g' zurückliefert, in dem eine Kante (a,b) genau dann exisitert, wenn eine Kante (b,a) in g existiert.

Ein Beispiel Graph g und sein inverser Graph g' sind unten dargetstellt.

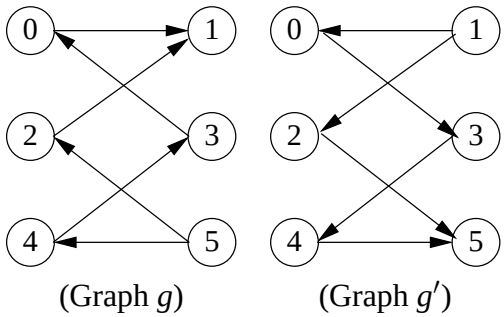


Abbildung 1: Graph g und sein inverser Graph g'

Aufgabe 4 Bipartiter Graph **(10 Punkte)**

Ein gerichteter Graph heißt bipartit, falls sich seine Knoten in zwei disjunkte Teilmengen A und B aufteilen lassen, so dass jede Kante des Graphen einen Knoten aus A mit einem Knoten aus B verbindet (d.h. für jede Kante (u,v) gilt entweder $(u,v) \in A \times B$ oder $(u,v) \in B \times A$). Definieren Sie eine Funktion `makeBipartition`, die als Argument einen bipartiten Graphen erhält und zwei disjunkte Teilmengen wie in der Definition (als ein Ocaml-Paar von Listen) zurückliefert. Erläutern Sie kurz Ihre Lösung.

Gehen Sie dabei davon aus, dass zu jeder Kante (u,v) des Graphen auch eine Gegenkante (v,u) existiert. Nutzen Sie den in der vorherigen Aufgabe angegebenen Daten-Typ zur Respräsentation gerichteter Graphen.

Zum Beispiel ist `([0;2;4],[1;3;5])` ein gültiger Rückgabewert für den in Abbildung 2 dargetsellten bipartiten Graphen g .

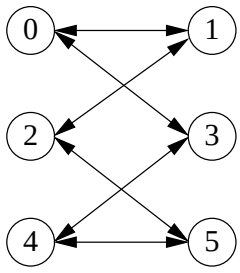


Abbildung 2: Bipartiter Graph

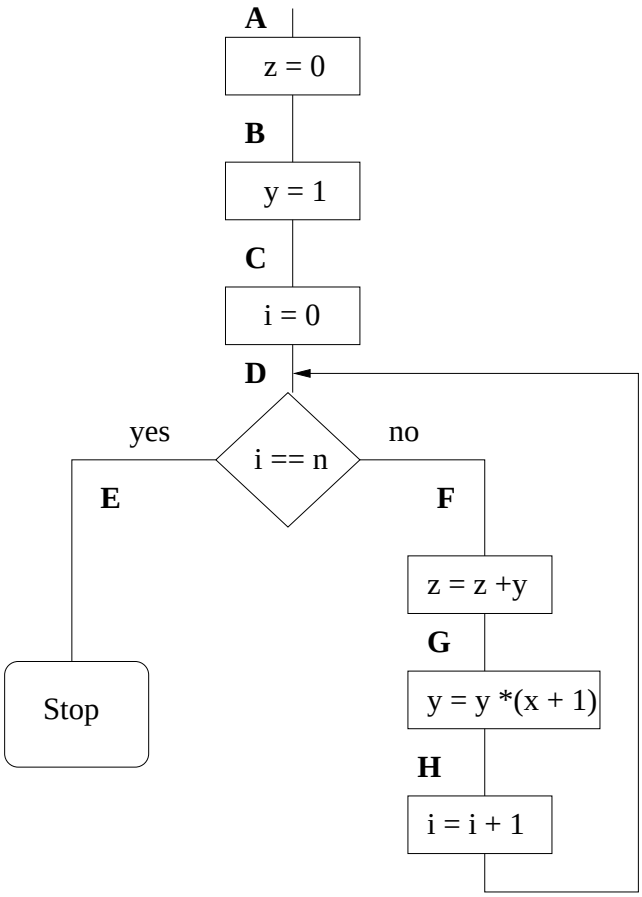
Aufgabe 5 Verifikation eines Mini-Java-Programms **(10 Punkte)**

Nebenstehend ist der Ausschnitt des Kontrollflußgraphen eines Programms mit den globalen und ganzzahligen Variablen `i, n, x, y, z` gegeben. Gegeben sind die Zusicherungen

$$A \equiv true$$

und

$$E \equiv y = (1+x)^n \wedge x \cdot z = y - 1$$



Geben Sie Zusicherungen `B, C, D, F, G, H` an und zeigen Sie lokale Konsistenz. Gegebenfalls durchzuführende Äquivalenzumformungen, Abschwächungen oder Verstärkungen sind zu erläutern!

Aufgabe 6 Verifikation funktionaler Programme **(10 Punkte)**

Gegeben sei folgende MiniOCaml-Funktionsdefinition:

```
let rec flip =
  fun list ->
    match list with
        [] -> []
      | (x,y)::tail -> (y,x)::(flip tail)
```

Zeigen Sie, dass die Funktion `flip` selbst-invers ist, d.h. es gilt: `flip (flip list) = list`