Universität Stuttgart
Fakultät 5, Institut FMI
EfidI 2 Zusatzkurs

SS 2005
Wolfgang Schmid
4. Juli 2005
Bearbeitungszeit 120 min

Testprüfung (so könnte ein Teil der Prüfung aussehen)

Zur Beachtung: Diese Testprüfung bezieht sich nur auf etwa 50% des Stoffes von Informatik II. Mindestens 50% der Prüfung Einführung in die Informatik I, II besteht aus Programmierung. Jeder Punkt wird mit 2 Minuten Arbeitszeit gleichgesetzt. Damit ist die Testklausur mit knapp 2 Stunden Bearbeitungszeit aufwendiger als die Prüfung. Sie ist auch keine Themengarantie oder Gewichtungsgarantie.

Aufgabe 1 Konfiguration einer Turingmaschine (ehemalige Prüfungsaufgabe) (2 + 2 = 4 P)

Gegeben sei eine deterministische Turingmaschine $M = (Q, \Sigma, \Gamma, q_0, \delta, F)$ mit $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $\Gamma = \Sigma \cup \{B\}$, $F = \{q_2\}$ und $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ wie folgt:

δ	0	1	B
q_0	$(q_1, 0, R)$	$(q_1, 1, R)$	$(q_2, 0, N)$
q_1	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_2, 1, N)$
q_2	–	–	–

Ein waagerechter Strich bedeutet, dass die Turingmaschine stoppt, d. h. $\delta(q, a) = (q, a, N)$ für $q \in Q$ und $a \in \Gamma$.

a) Geben Sie die vollständige Konfigurationsfolge der Maschine M angesetzt auf die Eingabe 11001 an.

b) Beschreiben Sie in Worten die von M berechnete Funktion.

Aufgabe 2 Aussagen über asymptotisches Verhalten (4 + 3 = 7 P)

Gegeben seien folgende Prozeduren bei gegebener globaler Variable n und Prozedur E:

```
procedure p1 is begin
  for i in 1.. n * n loop
    if ( n mod 2 = 0 ) or ( n mod 3 = 0 ) then
      for j in 1.. n loop
        E ;
      end loop ;
    else
      E ;
    endif ;
  end loop ;
end p1;
```

```
procedure p2 is begin
  for i in 1.. n * n loop
    if ( n mod 3 = 0 ) then
      for j in 1.. n loop
        E ;
      end loop ;
    else
      E ;
    endif ;
  end loop ;
end p2;
```

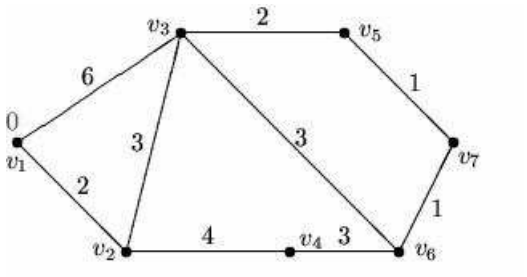
```
procedure p3 is begin
  for i in 1.. n * n loop
    if ( n mod 2 = 0 ) then
      for j in 1.. n loop
        E ;
      end loop ;
    else
      E ;
    endif ;
  end loop ;
end p3;
```

Sei $f_i : \mathbb{N}_0 \to \mathbb{N}_0$ mit $f_i(n) =$ die Anzahl der Aufrufe von E in Prozedur p_i für $i = 1, 2, 3$.

(a) Geben Sie die Funktionen f_i (i=1, 2, 3) an.
(b) Für welche Paare i,j gilt $f_i \in f_j$ (i, j = 1, 2, 3)?

Aufgabe 3 Anwendung des Algorithmus von Dijkstra (2 P)

Bestimmen Sie mit dem Algorithmus von Dijkstra einen Kürzeste-Wege-Baum vom Knoten v_1 aus. Die einzelnen Schritte sollen nachvollziehbar sein.



Aufgabe 4 (Semantik) (10 P)

```
function exp(x,y:natural):natural;
var Erg:natural;
begin Erg:= 1;
  while y>0 loop
    if   y mod 2 = 0 then
      y:=y div 2;
      x:=x*x;
    elsif
      y:=y-1;
      Erg:=Erg*x;
    end if;
  end loop;
  return(Erg);
end;
```

Beweisen Sie mit geeigneten Zusicherungen und den Hoareschen Regeln, dass die Funktion exp x^y berechnet.

Aufgabe 5 (Semantik) (5 P)

Leiten Sie die *schwächste Vorbedingung* (*weakest precondition)* wp aus der entsprechenden *Nachbedingung* (*postcondition*) für folgendes Programmstück ab:

```
a, b: INTEGER
a := 2 * b - 2;
if a < 16 then
  b := 2 * a - 16;
else
  b := a+8;
end if;
b := b + 4;
{ Q ≡ 8 ≤ b ≤ 20 }
```

Aufgabe 6: (Einfügen in einen Suchbaum) (2 P)

Zeichnen Sie den Graphen eines zu Anfang leeren Suchbaumes, in den nacheinander die folgenden Elemente eingefügt werden
3, 1, 7, 2, 12, 6, 4, 10, 8, 5, 11, 9

Aufgabe 7: (Optimaler binärer Suchbaum) (3 P)

Gegeben seien die Schlüssel k_i = A, B, C, D mit ihren Suchhäufigkeiten p_i = 3, 5, 2, 4 (i=1,..,4). Konstruieren Sie einen optimalen binären Suchbaum nach dem in der Vorlesung besprochenen Verfahren und geben Sie dabei alle notwendigen Zwischenschritte an.

Aufgabe 8 (**AVL-Bäume**) (4 + 1 = 5 P)

a) Fügen Sie sukzessiv die Schlüssel 10, 20, 30, 40, 50, 60 und 70 in einen anfangs leeren AVL-Baum ein. Löschen Sie sukzessiv die Schlüssel 10, 20 und 40 aus dem Baum. Beschreiben Sie Ihr Vorgehen und geben Sie alle Zwischenergebnisse an.

b) Gegeben sei ein AVL-Baum mit 10^5 Knoten, die vollständig im Hauptspeicher Platz finden, und mit einer Verarbeitungszeit von 10^{-6} Sekunden pro Vergleich. Wie lang dauert die erfolgreiche Suche nach einem Schlüssel im schlechtesten Fall (Angabe des Berechnungstermes genügt)?

Aufgabe 9 (Aufbau von B-Bäumen) (3 + 2 = 5 P)

(Aufgabenteil a ist etwas zu umfangreich für die Prüfung)
a) Konstruieren Sie einen (zu Beginn leeren) B-Baum der Ordnung 2 durch Einfügen der folgenden Schlüssel (Angabe des Endbaumes genügt):

48, 23, 35, 66, 7, 3, 71, 12, 55, 2, 1, 9, 10, 25, 39, 42, 91, 84, 74.

b) Löschen Sie in dem entstandenen B-Baum sukzessive folgende Schlüssel: 35, 9, 55 (jeder Wert soll aus dem Originalbaum gelöscht werden).

Aufgabe 10 (Hashing) (3 + 2 = 5 P)

Wie sieht eine Hashtabelle der Größe 11 aus, wenn nacheinander die Werte

15, 38, 21, 26, 39, 37, 16, 6

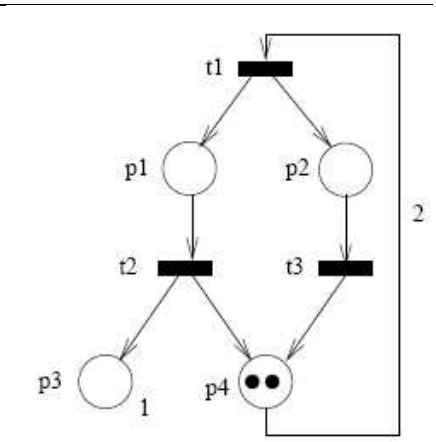
eingefügt und danach die Werte 26 und 6 gelöscht werden? Der Zugriff soll mittels linearem Sondieren mit der folgenden Hashfunktion erfolgen: $f(x) = x \bmod 11$.

Aufgabe 11: (3 Phasen mischen) (3 P)

Sortieren Sie die Elemente 8,7,6,5,4,3,2,1 mit 3 Phasenmischen. Geben Sie die Zwischenstände nach jedem Mischen an. Wieviele Elemente muss ein Feld haben, damit 3 Phasenmischen besonders effizient ist?

Aufgabe 12 (Petri-Netz Modell 5 P)

Gegeben sei das nebenstehende Petri-Netz. Beschreiben sie das Netz durch Angabe der Stellenmenge *S*, der Transitionenmenge *T*, der Markierung M_0 sowie der Flussrelation *F*. Geben Sie die Kantengewichte *W* und die Kapazitäten *K* der Stellen an. Geben Sie die Vor- und Nachbereiche der Transitionen und Stellen an. Welche Transitionen sind aktiviert?



Bestimmen Sie die Erreichbarkeitsmenge M_0. Ist das Netz deadlockfrei? Untersuchen Sie das Netz auf Lebendigkeit. Geben Sie alle echten S und T-Invarianten an.

(Zusatzaufgabe) Aufgabe 13 (Rehashing 8 P)

Was muss man tun, wenn beim Hashing der Auslastungsgrad über 80% hinausgeht oder gar den Wert 1 erreicht? Dann muss man die Hashtabelle verlängern, also *p* durch eine Zahl $q > p$ ersetzen, hierfür eine neue Hashfunktion festlegen und die neue Hashtabelle aus der alten Tabelle, in der die Schlüssel in den Plätzen von 0 bis p-1 standen, errechnen.

Diesen Vorgang der Umorganisation innerhalb der bestehenden Hashtabelle bezeichnen wir als "Rehashing". Dieses Verfahren wird auch verwendet, wenn man Schlüssel, statt sie zu löschen, nur als "gelöscht" markiert, wodurch im Laufe der Zeit der Auslastungsgrad zu groß wird und eine Umorganisation mit dem gleichen *p* notwendig wird (Rehashing mit $p = q$). In der Hashtabelle steht bei jedem Eintrag ein Bit zur Verfügung.

Finden Sie eine geeignete Datenstruktur für die Hashtabelle wenn die Einträge vom Typ integer sind. Programmieren Sie den Algorithmus des Rehashings bei global bekannter Hashtabelle und global bekannten Werten *p* und *q*. Als Hashfunktion verwenden Sie $h(k) = k \bmod q$ und als Kollisionsstrategie lineares Sondieren mit c=1.

Allgemeine Hinweise:

- Bei weiteren Fragen, wenden Sie sich bitte an W. Schmid (sltsoftware@yahoo.de).
- Weitere Hinweise finden Sie auf unserer Veranstaltungswebseite unter:
  http://www.info2.de.vu
  http://www.zusatzkurs.de.vu