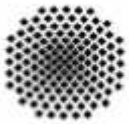
Universität Stuttgart
Fakultät 5, Institut FMI
EfidI 1 Zusatzkurs



WS 2004/2005
Dowertill + Schmid
3. März 2005
Bearbeitungszeit 120 min

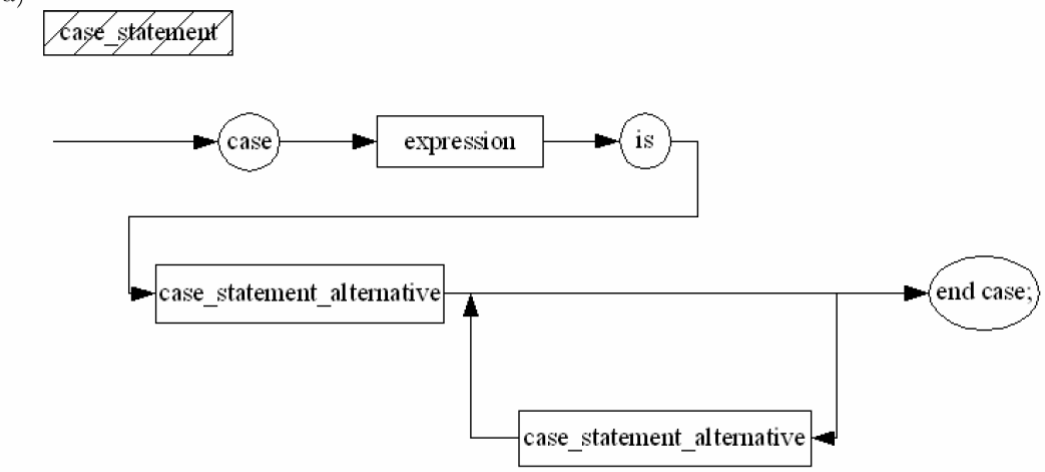
Testprüfung (so könnte ein Teil der Prüfung aussehen)

Zur Beachtung: Diese Testprüfung bezieht sich nur auf den Stoff von Informatik I. Mindestens 50% der Prüfung Einführung in die Informatik I, II besteht aus Programmierung. Jeder Punkt wird mit 2 Minuten Arbeitszeit gleichgesetzt. Sie ist auch keine Themengarantie oder Gewichtungsgarantie.

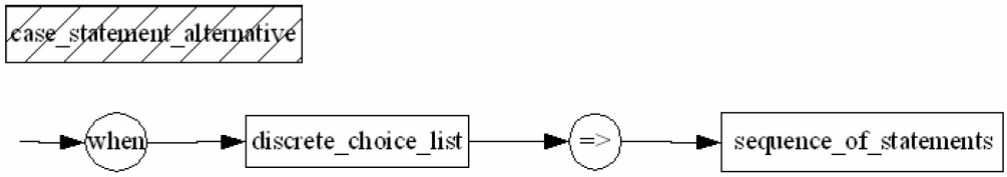
Aufgabe 1 (EBNF 2P)

Erstellen sie aus den Syntaxdiagrammen die dazugehörigen EBNFen.

a)



b)



Aufgabe 2 (Zahlensysteme 2 P)

Ergänzen sie die Tabelle mit den richtigen Werten:

Binärsystem	Oktalsystem	Dezimalsystem	Hexadezimalsystem
00111010			
	255		
		69	
			C3

Aufgabe 3 (Rekursion 4 P)

Schreiben Sie eine rekursive Funktion zur Berechnung der *n*-ten Potenz einer reellen Zahl *x* mit *x* vom Typ FLOAT und *n* vom Typ INTEGER. Um die Zahl der Multiplikationen gering zu halten, sind folgende Zusammenhänge zu verwenden:

$$x^n = \frac{1}{x^{-n}} \text{ für } n < 0$$
$$x^n = x^{n/2} \cdot x^{n/2} \text{ für gerade } n$$
$$x^n = x^{(n-1)/2} \cdot x^{(n-1)/2} \cdot x \text{ für ungerade } n$$

Aufgabe 4 (Binäre Suche 2 + 1 = 3 P)

a) Gegeben sei ein Array, auf dem der Wert 17 mittels binärer Suche gesucht werden soll. Zeichnen Sie die einzelnen Schritte des Suchvorgangs, indem Sie mit Pfeilen die Sprünge von dem gegenwärtigen Element auf das nächste Element markieren.

1	3	4	7	9	10	11	13	17	24	25	29	31	42

b) Welchen Aufwand hat die binäre Suche im
- best case:
- worst case:

Aufgabe 5 (Bäume 2 P)

Wie viele Bäume lassen sich aus einem gegebenen Binärbaum mit *n* Knoten durch Anhängen eines neuen Knotens konstruieren? Verallgemeinern Sie Ihr gefundenes Ergebnis auf Bäume vom Grad *m*. (Induktion ist nicht verlangt)

Aufgabe 6 (Parameterübergabe 6 P)

Analysieren Sie das folgende Ada-Programm. Welche Ausgabe erzeugt es?

```
with Ada.Text_IO, Ada.Integer_Text_IO;
use  Ada.Text_IO, Ada.Integer_Text_IO;

procedure Scope is

a, b, c: Integer;

procedure Write3Int(x, y, z: Integer) is
begin
  Put(x,3);
  Put(y,3);
  Put(z,3);
  New_Line;
end Write3Int;

procedure P1(b: in out Integer; d,c: Integer) is
begin
  a:= (c + a) * d;
  b:= a + c - d;
  Write3Int(a,b,c);
end P1;

procedure P2(a: in out Integer; c,d: Integer) is
begin
  a:= c + d;
  b:= a * c;
  Write3Int(a,b,c);
end P2;

begin -- Scope
  a:=1; b:=2; c:=3;
  Write3Int(a,b,c);
  P1(a, b, c);
  Write3Int(a,b,c);
  P2(c, c, b);
  Write3Int(a,b,c);
end Scope;
```

Aufgabe 7 (O Notation 3 P)

Ordnen Sie folgende Funktionen bezogen auf die O-Notation und begründen Sie Ihre Wahl:

$$f_1(n) = \begin{cases} n & \text{falls } n \text{ ungerade} \\ 3 * n^3 & \text{sonst} \end{cases} \qquad f_2(n) = \begin{cases} 3 * n^3 & \text{falls } n \text{ ungerade} \\ n^2 & \text{sonst} \end{cases}$$

$$f_3(n) = 2^n + 3^n \qquad f_4(n) = \sqrt{\tfrac{n}{2}}$$

Aufgabe 8 (Programmanaylse 2 P)

Bestimmen Sie das asymptotische Laufzeitverhalten des folgenden Programmstücks.

```
procedure m2(in n:integer) is
summe, i :integer;
begin
  summe:= 0; i:=1;
  while(i <= 2*n*n) loop
    summe = summe + 1;
    i = i + 1;
  end loop;
end m2;
```

Aufgabe 9 (Grammatik 2 + 2 + 2 + 1 = 7 P)

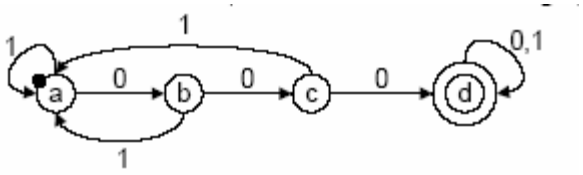
Gegeben sind folgende formale Sprachen:

$$L_1 = \{ a^n b^n a^m b^m \mid n \geq 0, m \geq 0 \}$$
$$L_2 = \{ a^n b^m a^m b^n \mid n \geq 0, m \geq 0 \}$$
$$L_3 = \{ a^2 b^n a^2 b^n \mid n \geq 0 \}$$

a) Geben Sie eine Grammatik an die L_1 erzeugt.
b) Geben Sie eine Grammatik an die L_2 erzeugt.
c) Geben Sie eine Grammatik für die Sprache $L_1 \cap L_2 \cap L_3$ an.
d) Geben Sie eine Grammatik für die Sprache $L_1 \cup L_2$ an.

Aufgabe 10 (Analyse eines DFA 1 + 2+ 1 = 4 P)

Es sei der folgende deterministische, endliche Automat A gegeben:



Stellen Sie die DFA formal (als Quintupel) dar.Welche Zeichenketten (Worte) akzeptiert der Automat A? Geben Sie eine Grammatik an, welche die gleiche Sprache erzeugt.

Aufgabe 11 Mensch ärgere Dich nicht (15 P)

An dieser Stelle gehe ich davon aus, dass Sie das Spiel kennen – wenn Sie dies nicht tun, wird der folgende Aufgabenteil schwierig. In einer Klausur würde das Spiel (hoffentlich) besser erläutert. Eine online Version des Spieles finden Sie unter http://pc-anfaenger.de/sam/madn/madn3.htm

Mensch ärgere Dich nicht ist ein deutsches Gesellschaftsspiel für 2-6 Personen. Es zählt zu den deutschen Klassikern unter den Brettspielen. Ziel von *Mensch ärgere Dich nicht* ist es, alle vier eigenen Spielfiguren (Pöppel) von den Startfeldern auf die Zielfelder zu ziehen. Dabei müssen die Figuren das Spielbrett einmal umrunden. Über die Anzahl der zu ziehenden Felder pro Runde entscheidet ein Würfel.

Das Spielfeld hat 40 Felder. Spieler 1 setzt seine Figuren vom Vorrat immer auf Feld 0, Spieler 2 auf Feld 10, Spieler 3 auf Feld 20 und Spieler 4 auf Feld 30. Würfelt Spieler 4 auf dem Feld 38 eine 5, so setzt er seine Figur auf das Feld (38+5) mod 40 = 3.

Erreicht ein Spieler mit einer Figur *A* ein Feld *F*, welches von einer fremden Figur *B* besetzt ist, so muss er diese Figur ‚schlagen' – *A* bleibt auf *F* stehen, *B* muss in den Vorrat des Figureigentümers zurück. Eigene Figuren können nicht geschlagen werden. Ein Zug auf ein Feld, welches von einer eigenen Figur besetzt ist, ist unmöglich.

Um das Spiel in endlicher Zeit programmierbar zu machen, führen wir einige Sonderregeln ein:

1) Eine Spielfigur darf nach jedem Wurf vom Vorrat auf das Spielfeld gesetzt werden (die ‚Sechserregel' entfällt).

2) Das Spielfeld gilt mit dem Erreichen oder Überschreiten des Startfeldes als umrundet. Die Spielfigur wird sofort aus dem Spiel genommen und kann eventuelle Figuren auf dem erreichten Feld nicht schlagen.

3) Es müssen 4 Spieler teilnehmen.

4) Nach einer 6 wird nicht nochmals gewürfelt.

Schreiben Sie ein Simulationsprogramm für das Spiel. Entwerfen Sie zunächst geeignete Datenstrukturen. Ein Zug eines Spielers soll folgendermaßen aussehen: Zuerst soll das Programm würfeln (random), dann dem Spieler zeigen, welche seiner vier Figuren er ziehen kann (ein Zug ist immer möglich) und mit welcher er schlagen kann. Der Spieler hat dann die Auswahl, welche Figur er zieht. Falsche Eingaben sind abzufangen. Danach kommt der nächste Spieler an die Reihe. Das Spiel soll enden, wenn ein Spieler mit allen Figuren das Spielfeld ein Mal umrundet hat. Das Programm soll die komplette Verwaltung des Spieles übernehmen.

Aufgabe 12 Listenumordnung (3 P)

Schreiben Sie eine Prozedur, welche eine gegebene Liste umordnet, so dass alle Knoten mit geradem Wert *val* am Anfang der Liste stehen. Die Ordnung innerhalb der geraden Knoten spielt keine Rolle.

Die Prozedur soll die folgende Schnittstelle haben:

```
type  PNodeDesc;


type   NodeDesc is record
   val: integer;
   next: PNodeDesc;
end record;


type  PNodeDesc  is access NodeDesc;
```

Aufgabe 13 (Quersumme rekursiv 2P)

Implementieren Sie eine rekursive function *Quer*, die die Quersumme einer natürlichen Zahl berechnet.

Aufgabe 14 (Backtracking 5 P)

Schreiben Sie ein Programm in einer beliebigen Programmiersprache, welches unter Verwendung von Backtracking alle n - dimensionalen Vektoren mit Komponenten aus {0,1} erzeugt, welche die Eigenschaft besitzen, dass maximal drei aufeinander folgende Komponenten identisch sind. Beispiel für n=6: (0, 0, 0, 1, 1, 0) ist zulässig, (1, 0, 0, 0, 0, 1) dagegen nicht.

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