Klausur zu Programmieren I

WS 93 / 94

9. Februar 1994

Name _____

Vorname _____

Matrikel-Nr. _____

Aufgabe		Punkte	Bewertung
1	a)	2	
	b)	2	
	c)	2	
	d)	2	
	e)	2	
	f)	2	
2		25	
3		8	
4	a)	2	
	b)	5	
	c)	8	
5		10	
6		10	
7		20	
Summe		**100**	

Aufgabe 1

(*a*) Wieviele Kataloge befinden sich in Ihrem Heimat-Katalog, deren Name nicht mit einem Punkt (`.`) beginnt? Schreiben Sie die UNIX-Befehlszeile hin.
Hinweis: Sie befinden sich nicht unbedingt in Ihrem Heimat-Katalog.

(*b*) Sie besitzen zwei Dateien *kapitel.1* und *kapitel.2*, wobei *kapitel.2* die Fortsetzung von *kapitel.1* ist. Geben Sie zwei Wege an, *kapitel.2* so an *kapitel.1* anzufügen, daß *kapitel.1* alles enthält.

(i) auf Shell-Ebene

(ii) via **vi**

(*c*) Geben Sie ein UNIX-Kommando an, das die Anzahl der Leerzeilen in der Datei *kapitel.1* ausgibt.

(*d*) Geben Sie zwei Wege an, wie Sie Ihr Modula-2-Programm *doit.m2* zu einer ausführbaren Datei namens *doit* machen können.

(*e*) Sie verwenden in Ihrem Modula-2-Programm folgende Fehlerausgänge:

```
IF error1 THEN Exit(1) END;
IF error2 THEN Exit(2) END;
IF error3 THEN Exit(3) END;
```

Angenommen das Programm terminiert bei der Ausführung wegen eines Fehlers. Wie können Sie ohne das Quellprogramm zu studieren, feststellen, ob *error1*, *error2* oder *error3* der Grund dafür war ?

(*f*) Sie sind vielleicht schreibfaul und schreiben beim Editieren mittels **vi** anstatt `PROCEDURE` nur kurz `PROC`. Am Ende ersetzen Sie alle Vorkommen von `PROC` durch `PROCEDURE`. Geben Sie das entsprechende **vi**-Kommando an.

Aufgabe 2

Schreiben Sie ein Modula-2-Programm namens **head**, das jeweils den Dateinamen und die ersten paar Zeilen der als Argumente angegebenen Dateinamen auf die Standardausgabe ausgibt. Die Anzahl der auszugebenden Zeilen ist auf fünf Zeilen vordefiniert, kann jedoch mittels einer Option geändert werden.
Aufruf:

```
$ head [-nnn] file { file }
```

Ist ein Dateiargument keine Datei, so soll ein entsprechender Hinweis auf die Diagnose-Ausgabe gegeben werden. Die nach der UNIX-Konvention übliche Behandlung der Standardeingabe — Dateiargument ist ein Minuszeichen — muß nicht berücksichtigt werden. Folgende Prozedur können Sie als gegeben voraussetzen:

```
(*
 * PROCEDURE FreadLine(f : FILE; VAR line : ARRAY OF CHAR): BOOLEAN;
*)

FROM Gegeben IMPORT FreadLine;
```

Die Prozedur **FreadLine** liest aus der mit dem FILE-Pointer **f** verbundenen Datei Zeichen bis zum *newline* in das Array **line** ein. Falls die Dimension des Arrays nicht ausreicht, wird trotzdem bis zum *newline* gelesen. Ansonsten wird das *newline* (12C) durch ein *Nullbyte* (0C) ersetzt. Das Resultat der Funktionsprozedur ist FALSE, falls das Dateiende erreicht oder ein Lesefehler erfolgt ist, ansonsten TRUE.

Aufgabe 3

Geben Sie die Ausgabe des folgenden Modula-2 Programms an.
Begründen Sie schrittweise die Zwischenergebnisse.

```
MODULE obscure;

   FROM InOut IMPORT WriteCard, WriteLn;

   VAR x, y : CARDINAL;

   PROCEDURE Obscure(VAR x : CARDINAL; y : CARDINAL): CARDINAL;
   BEGIN
      INC(x); RETURN (x + y)
   END Obscure;

BEGIN
   x := 4; y := 2;

   IF (Obscure(x, y) < Obscure(y, x)) OR (Obscure(x, y) > Obscure(y, x)) THEN
      x := Obscure(x, y); y := Obscure(y, x)
   ELSE
      x := Obscure(y, x); y := Obscure(x, y)
   END;

   WriteCard(x, 0); WriteLn;
   WriteCard(y, 0); WriteLn
END obscure.
```

Aufgabe 4

Sie implementieren ein Grafikprogramm zur Darstellung und Verwaltung von einfachen geometrischen Objekten. Zugriffsfunktionen wie Zeichnen, Einfügen, Ändern und Löschen von geometrischen Objekten sollen dabei auf einer allgemeinen Datenstruktur **Objekt** operieren. Jedes Objekt wird durch seine Geometrie und grafischen Attribute beschrieben. Die Geometrie eines Objekts wird durch Angabe von kartesischen oder polaren Koordinaten spezifiziert, um eine vom Ausgabemedium unabhängige Darstellung zu erreichen. Die grafischen Attribute gliedern sich in objektspezifische und allen Objekten gemeinsame Attribute. Die geometrischen Objekte sind *Punkte*, *Linien*, *Kreise* und zu den Koordinatenachsen parallele *Rechtecke*.

Die Geometrie der einzelnen Objekte ist in folgender Tabelle definiert:

Geometrisches Objekt	**Koordinaten** *(kartesisch oder polar)*
Punkt	Koordinaten des Punktes
Linie	Anfangs- und Endpunkt-Koordinaten
Rechteck	Koordinaten zweier gegenüberligender Punkte
Kreis	Mittelpunkt-Koordinaten und Radius

Allgemeine und spezifische Attribute sind in nachfolgender Tabelle definiert.
Die möglichen Attributwerte sind in Klammern angegeben.

Geometrisches Objekt	**Farbe** *(rot, grün oder blau)*	**Linienbreite** *(1 — 5)*	**Füllmodus** *(ja / nein)*
Punkt	×	—	—
Linie	×	×	—
Rechteck	×	×	×
Kreis	×	×	×

(*a*) Definieren Sie für die Attribute *Farbe*, *Linienbreite* und *Füllmodus* geeignete Datentypen in Modula-2.

(*b*) Definieren Sie einen geeigneten Datentyp **Koordinaten** mit Hilfe eines varianten Records in Modula-2. Abszisse und Ordinate bzw. Radius und Polarwinkel seien reellwertig.

(*c*) Definieren Sie unter Verwendung von Teilaufgabe (a) und (b) einen geeigneten Datentyp **Objekt** mit Hilfe eines varianten Records in Modula-2.

Aufgabe 5

Eine (fiktive) UNIX-Shell erlaube folgende Kommandozeilen-Syntax:

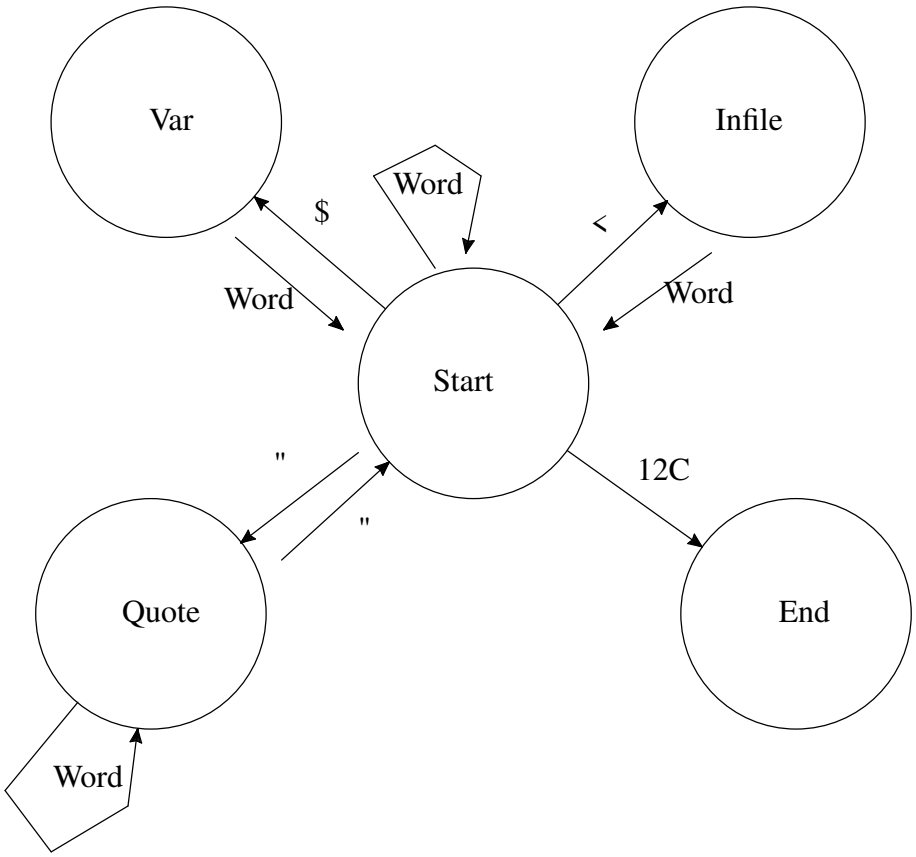
```
CmdLine   ::= SingleCmd { '|' SingleCmd } [ '>' File ] '12C'
SingleCmd ::= Word { Word }
File      ::= Word
```

Die in einfache Hochkommas eingeschlossenen Symbole sind Terminalsymbole. Das Symbol *Word* ist zwar aus einzelnen Zeichen aufgebaut, soll hier aber nicht weiter erläutert werden. Betrachten Sie also *Word* als Terminalsymbol.

Geben Sie einen endlichen Automaten in grafischer Darstellung an (vgl. Aufgabe 6), der diese Sprache erkennt.

Aufgabe 6

Gegeben sei der folgende endliche Automat:



Beschreiben Sie die von diesem Automaten definierte Sprache in EBNF.

Aufgabe 7

Gegeben seien folgende Vereinbarungen:

```
CONST Length = 20;
      Size   = 3329;

TYPE  Word = ARRAY [0..Length] OF CHAR;

      Entry = RECORD
         occupied : BOOLEAN;
         element  : Word;
      END;

      Table = ARRAY [1..Size] OF Entry;

VAR   table : Table;
```

Schreiben Sie eine Prozedur **Insert** (Prozedur-Kopf s.u.), die Worte vom Typ *Word* in die Tabelle *table* vom Typ *Table* einträgt. Der Index in der Tabelle soll aus den Zeichen des Wortes berechnet werden, und zwar so, daß alle Zeichen des Wortes in diese Berechnung eingehen und alle Indexpositionen grundsätzlich erreichbar sind. Ist die Indexposition bereits belegt (`occupied = TRUE`), so soll auf die Diagnose-Ausgabe ein Hinweis ausgegeben werden.

Der Prozedurkopf:

```
PROCEDURE Insert(VAR t : Table; w : Word): BOOLEAN;
(*liefert TRUE, falls eingefuegt; FALSE, falls bereits besetzt*)
```

Hinweis: Die Transformation eines Wortes (Buchstaben $x_1, x_2, \cdots, x_n$) soll auf der Basis folgender Abbildungsvorschrift erfolgen:

$$[x_1, x_2, \cdots, x_n] \rightarrow \sum_{i=1}^{n} g(x_i) \cdot 10^i$$

$g(x_i)$ ist dabei die übliche Transformationsfunktion zur Umwandlung von `CHAR` in `CARDINAL`.