Klausur -Lösung zu UDB I

WS 97

05. Februar 1998

Name ______________________________

Vorname ______________________________

Matrikel-Nr. ______________________________

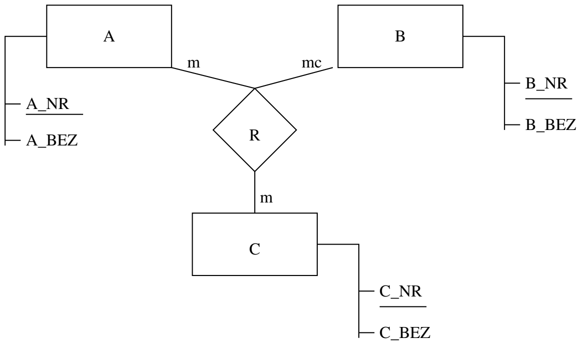
Studienfach ______________________________

Aufgabe		Punkte	Bewertung
1	a)	3	
	b)	3	
	c)	2	
	d)	2	
2	a)	2	
	b)	6	
	c)	4	
3	a)	2	
	b)	10	
4		10	
5	a)	4	
	b)	4	
6	a)	6	
	b)	4	
7		8	
8		10	
Summe		**80**	

Hinweis: Bitte benützen Sie den jeweils vorgesehenen Platz. Sollte dieser nicht ausreichen, verwenden Sie bitte die Rückseite und machen Sie eine eindeutige Markierung!

Aufgabe 1

Gegeben sei folgendes (abstrakte) ER-Diagramm:



Wertebereiche:

A_NR	A_BEZ	B_NR	B_BEZ	C_NR	C_BEZ
{1,2,...10}	{a1, a2,...,a5}	{1,2,...,20}	{b1,b2,...b25}	{1,2,3,4,5}	{c1,c2,...,c9}

(*a*) 3 Punkte

Geben Sie in Skizzenform an, wie dieses in Tabellen umgesetzt werden kann!

A	A-NR	A_BEZ
(b)	1	a1

B	B_NR	B_BEZ

C	C_NR	C_BEZ

R	A_NR	B_NR	C_NR
(d)	1	1	1

(*b*) 3 Punkte

Tragen Sie einige wenige Datensätze so ein, daß eine Verletzung der durch die Komplexitätsgrade definierte Integrität entsteht!

(*c*) 2 Punkte

Wieviele Entities können im Entity-Typ **A** maximal vorkommen?

Anzahl = 10

(*d*) 2 Punkte

Tragen Sie weitere (möglichst wenige Datensätze oben ein, so daß eine Verletzung der Beziehungsintegrität entsteht (diese sind zur Unterscheidung von Teilaufgabe b) deutlich zu markieren)!

Aufgabe 2

In Tcl-Skripten gibt es immer wieder einmal die Notwendigkeit, eine temporäre Datei anzulegen.

(*a*) 2 Punkte

Wo und wie legen Sie eine solche Datei an? Bitte geben Sie auch dazu an, warum Sie es so wie beschrieben machen!

Im Katalog **/tmp** - string mit angefügter Prozeß-ID; dieser Katalog kann vollständig gelöscht werden; Probleme gibt es, wenn Prozesse zwar noch aktiv, aber keinen (oder momentan) keinen Filedeskriptor auf ihre tempor*re Datei haben

(*b*) 6 Punkte

Schreiben Sie eine Tcl-Prozedur, die eine temporäre Datei anliegt und einen “Zugriffsweg” darauf zurückliefert. Diese temporäre Datei soll allerdings wieder entfernt werden, egal wie das diese Prozedur aufrufende Skript terminiert!

```
proc mktemp { mode } {
   set tempfile /tmp/tcl[pid]
   set f [open $tempfile $mode ]
   # aufraeumen: link auf tempfile entfernen
   exec rm $tempfile
   return $f
}
```

(*c*) 4 Punkte Unter Benutzung dieser Prozedur öffnen Sie diese Datei jetzt bitte, schreiben das Wort "Hallo" hinein, lesen es anschließend wieder und geben Sie das Gelesene auf die Diagnoseausgabe aus!

```
set fd [mktemp w+]
puts $fd "Hallo"
seek $fd 0 start
gets $fd line
puts stderr "Gelesen: $line"
```

Aufgabe 3

Sie haben sich entschlossen, in Ihrer (anstehenden?) Diplomarbeit Literaturzitate in eckige Klammern zu setzen. Das sieht dann so aus: *"Dies ist ein kleiner Text frei nach [Grabert98] und [Melzer98]. Sie beschreiben darin ihre Situation an der SAI, wie sie sie aus dem Text von [Schweiggert96] kennengelernt hatten."*

Sie können davon ausgehen, daß eckige Klammern immer Zitate enthalten, allerdings kann nach der schließenden Klammer ein Interpunktionszeichen (**.,;:?!**) stehen
Nun möchten Sie eine Übersicht erhalten über Ihre Zitate.

(*a*) 2 Punkte
Dazu schreiben Sie zunächst für *egrep* einen regulären Ausdruck, der alle Zeilen herausfiltert, in den ein Zitat vorkommt.

```
egrep '\[.*\]' file
```

(*b*) 10 Punkte
Als nächstes schreiben Sie ein Tcl-Skript *getlit*, das als Argument die zu durchsuchende Textdatei erhält und auf die Standardausgabe alle Zitate ausgibt. Aufruf: **getlit file**. Ob die Datei existiert oder lesbar ist, muß nicht geprüft werden! Natürlich können und sollen Sie in Ihrem Skript das Kommando *egrep* nutzen!

```
#!/usr/local/bin/tclsh

if { $argc != 1 } {
     puts stderr "Usage: $argv0 file"
     exit 1
}

set file [lindex $argv 0]
set tempfile "TMP"
# suchen
catch { exec egrep {\[.*\]} $file >  $tempfile } exitstatus

if { $exitstatus > 0 } {
      # nichts enthalten oder Fehler bei egrep
      puts stderr "Kein Zitat gefunden oder Fehler bei egrep"
      exit 1
}

set f [open $tempfile r ]
while { [gets $f line] >= 0 } {
     set words [split $line]
     for { set i 0 } { $i < [llength $words] } { incr i } {
          set w [ lindex $words $i ]
          if { [regexp {(^\[.*\])[.,;:!?]?$} $w all s1] == 1 } {
               puts stdout $s1
          }
     }
}
```

Aufgabe 4

10 Punkte

Für einen Vortrag produzieren Sie eine stattliche Anzahl von Folien, die Sie jeweils einzeln in einer Datei (ASCII-Form) ablegen. Diese nennen Sie sinnigerweise *folie.1, folie.2, ..., folie.20, folie.21, ...* Dabei können in der Numerierung durchaus Lücken sein; im Lauf Ihres Schaffens sind natürlich auch Dateien wie folie11.old u.dgl. entstanden. In jeder Datei sind als Fußnote (hier nicht weiter wichtig) mit der Bezeichnung FOLIENNUMMER Platzhalter für die Foliennumerierung und mit der Bezeichnung FILE Platzhalter für den jeweiligen Dateinamen enthalten. Beide stehen in einer Zeile in der Form “FOLIENUMMER / FILE”.
Die *folie.12* sieht z.B. so aus:

```
Und dies meine Schlussbemerkung:

Sie predigen Wasser und trinken heimlich den Wein

copyright SAI / NR-FOLIENNUMMER / FILE
```

Weiterhin steht Ihnen ein Tool **makeps** zur Verfügung, mit denen Sie für jede Datei eine Postscript-Form (Dateiname = Dateiname + Extension ".ps") produzieren können. Dieses liest von *stdin* und schreibt nach *stdout*.

Schreiben Sie ein Shell-Skript (**bash**), das zum einen die Foliendateien in der durch die Extension bestimmten Reihenfolge (von 1 an lückenlos) durchnumeriert (Platzhalter FOLIENNUMMER), den Platzhalter FILE durch den jeweiligen Dateinamen ersetzt und dann mit dem Tool *makeps* die jeweilige Postskriptdatei erzeugt. Die größte “Extensions-Nummer” (>= 1) wird als Argument übergeben.

```
if [ $# != 1 ]
then
   echo "usage: $0 number"
   exit 1
fi

max_e_nr=$1

f_nr=1
e_nr=1

while [ $e_nr -le $max_e_nr ]
do
   file=folie.$e_nr
   if [ -f $file ]
   then
      cat $file | sed "s/FOLIENNUMMER \/ FILE/$f_nr \/ $file/" > $file.ps
      f_nr=`expr $f_nr + 1`
   fi
   e_nr=`expr $e_nr + 1`
done
```

Aufgabe 5

(*a*) 4 Punkte

Was versteht man im Unix-Kontext unter *Environment*? Geben Sie zwei Beispiele aus dem dessen Inhalt mit kurzer Erläuterung an! (Diese sollten sich auf den Unterrichtsrechner *turing* beziehen und in ihrer Bedeutung nicht benutzerspezifisch sein!)

Das Environment enthält die Definition einer Reihe von Variablen, auf die verschiedene Prozesse zugreifen.
Bsp.:
PATH: gibt den Suchpfad an für die Shell zur Ausführung von Kommandos
PS1, PS2: Definieren den Primär-/Sekundärprompt der Shell
USER: definiert den login-Namen des Besitzers des Prozesses
PRINTER: definiert den default-Drucker

(*b*) 4 Punkte

Sie haben in Ihrem Heimatkalalog in einem Unterkatalog *bin* viele eigene nützliche Kommandos gesammelt. Als Sie diese zum ersten Mal testen (u.a. das Kommando xxx), kommt nur eine lapidare Meldung der *bash*-Shell:

bash: xxx: command not found

Woran kann das liegen? Die Datei xxx existiert in Ihrem *bin*-Verzeichnis, die Rechte sind auch richtig gesetzt. Was müssen Sie tun, damit Sie Ihre Programme, egal in welchem Katalog Sie sind, einfach über Ihren Namen ausführen zu können? (Angabe dessen, was Sie dazu am Rechner eintippen müssen).

```
PATH=:$HOME/bin$PATH
```

oder

```
PATH=$PATH:$HOME/bin
```

Aufgabe 6

(*a*) 6 Punkte
Nachdem Sie nun schon so oft darüber geärgert haben, daß das Kommando **rm** Dateien so knallhart löscht, schreiben Sie Ihr eigenes **myrm**, das, wenn mehr als 2 Argumente angegeben sind, die angegebenen Dateien auf *stdout* ausgibt, erst nachfragt, ob wirklich alle zu löschen sind und erst bei expliziter Eingabe von ‘j’ oder ‘J’ tatsächlich löscht.

```
if [ $# -gt 2 ]
then
     echo "Folgende Dateien wollen Sie loeschen:"
     echo $@
     echo "Wirklich loeschen (>j< eingeben)? "
     read answer
     if [ "$answer" = "j" ] || [ "$answer" = "J" ]
     then
          rm $*
     fi
else
     rm $*
fi
```

(*b*) 4 Punkte
Schreiben Sie ein Programm, das seine Argumente in umgekehrter Reihenfolge ausgibt, wobei maximal 9 Argumente zulässig sein sollen. Es ist das Kommando **eval** zu verwenden!

```
anzahl=$#
if [ $anzahl -gt 9 ]
then
     echo "Zuviele ($anzahl) Argumente"
     exit 1
fi

while [ $anzahl -gt 0 ]
do
     eval echo \$$anzahl
     anzahl=`expr $anzahl - 1`
done
exit 0
```

Aufgabe 7

8 Punkte

Folgende Dateistruktur sei unterstellt:

```
FILE ::= HEADER { RECORD }
HEADER ::= ATTR {":" ATTR} "\n"
RECORD ::= VALUE {":" VALUE } "\n"
```

Natürlich sollen die Zahl der "VALUEs" und die Zahl der "ATTRibute" übereinstimmen. Jemand hat die Datei *db_file* gefüllt, ohne auf die genannte Übereinstimmung zu achten. Schreiben Sie ein kurzes Shell-Skript, das feststellt, ob Übereinstmmung (Exit-Status 0) oder eine Verletzung dieser Forderung (Exit-Status 1) vorliegt.
Hinweis: IFS

```
IFS=":"
read head
set $head
head_anzahl=$#

while read line
do
     set $line
     if [ $# -ne $head_anzahl ]
     then
          exit 1
     fi
done
exit 0
```

Aufgabe 8

10 Punkte

Gegeben sei wie in den Übungen die Datei *logins.all*. Die ersten fünf Zeilen seien hier noch einmal wiedergegeben:

```
login:Name:Vorname(n):PLZ:Ort:Strasse:Tel:Studiengang
star:Sternthaler:Uwe:89069:Ulm:Eselsberg 3:6888345:Wima
smitty:Schmid:Franz Xaver:89063:Ulm:Hoervelsingerweg 7:9667501:Info
sweety:Urbanoff:Urban:89069:Ulm:Eselsberg 3:6888345:Bio
miller:Miller:Josef:89081:Ulm:Hasenkopf 5::Info
```

Diese Datei sollen Sie jetzt in ein assoziatives Array einlesen (Index: *login*) und anschließend die Datensätze mit gleichem Stdudiengang "verketten", d.h. ein weiteres Array mit Index *Studiengang* aufbauen, dessen Einträge eine Liste der *login*'s sind, die denselben *Studiengang* haben.

```
#!/usr/local/bin/tclsh

set f [open "./logins.all" r ]

gets $f head

while { [gets $f line] >= 0 } {
     set linelist [ split $line ":" ]
     set arr([lindex $linelist 0]) $linelist
}

foreach ind [array names arr] {
     set st [lindex $arr($ind) [expr [llength $arr($ind)] -1 ] ]
     set stud($st) [lappend stud($st) $ind]
}
foreach ind [array names stud] {
     puts stderr "$ind : $stud($ind)"
}
```