Klausur zu UNIX-basierte Implementierung kleiner Datenbanken
11. Februar 2004 (WS 03/04)

Bearbeitungszeit: 120 Minuten
NICHT MIT BLEISTIFT SCHREIBEN!

Name:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Aufgabe	Punkte	Bewertung
1	**14**	
(a)	6	
(b)	6	
(c)	2	
2	**8**	
3	**5**	
4	**8**	

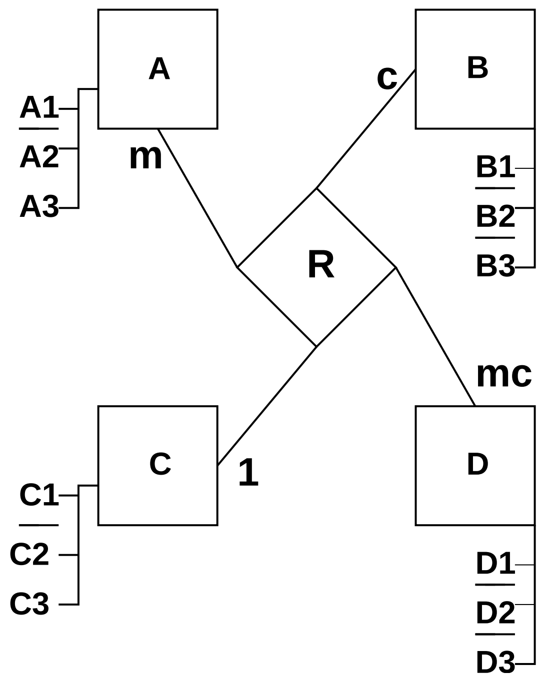
Aufgabe	Punkte	Bewertung
5	**6**	
6	**8**	
(a)	4	
(b)	4	
7	**8**	
8	**12**	
9	**4**	
Summe	**73**	

Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe oder die Rückseite unter Angabe der Aufgabe. Nennen Sie möglichst alle Annahmen, die Sie für die Lösung einer Aufgabe treffen! Alle Perl-Programme müssen ohne Warnungen bei Verwendung von `use strict;` und `-w` laufen.

Viel Erfolg!

Aufgabe 1 **(14 Punkte)**

Gegeben sei das folgende (abstrakte) ER-Diagramm gemäß der in der Vorlesung verwendeten Notation. Es sind nur die darin enthaltenen Informationen zum Sachverhalt gültig, darüber hinausgehende Annahmen dürfen nicht getroffen werden.



(a) 6 Punkte

Bewerten Sie die folgenden Aussagen in Bezug auf das ER-Diagramm. Falsche Antworten geben Abzug, negative Punkte werden nicht vergeben.

	Ja	Nein
Ist es zulässig, dass es $r_1 = (a', ., ., .), r_2 = (a'', ., ., .) \in R$ gibt mit $a', a'' \in A$, $r_1 \neq r_2$ sowie $a'.A1 = a''.A1, a'.A2 = a''.A2$	X	
Ist es zulässig, dass es $r_1 = (., b', ., .), r_2 = (., b'', ., .) \in R$ gibt mit $b', b'' \in B$, $r_1 \neq r_2$ sowie $b'.B1 = b''.B1, b'.B2 = b''.B2$		X
Ist es zulässig, dass es $r_1 = (., ., c', .), r_2 = (., ., c'', .) \in R$ gibt mit $c', c'' \in C$, $r_1 \neq r_2$ sowie $c'.C1 = c''.C1, c'.C2 = c''.C2$		X
Ist es zulässig, dass es $r_1 = (., ., ., d'), r_2 = (., ., ., d'') \in R$ gibt mit $d', d'' \in D$, $r_1 \neq r_2$ sowie $d'.D1 = d''.D1, d'.D2 = d''.D2$	X	
Ist es zulässig, dass es $a', a'' \in A$ gibt mit $a' \neq a''$ und $a'.A1 = a''.A1$		X
Ist es zulässig, dass es $b', b'' \in B$ gibt mit $b' \neq b''$ und $b'.B1 = b''.B1$	X	

(b) 6 Punkte

Geben Sie an, wie Sie dieses ER-Diagramm in Tabellen für eine relationale Datenbank umsetzen würden, so dass die Datenredundanz minimal ist. Primär- wie Fremdschlüssel müssen klar als solche kenntlich gemacht werden.

A	<u>A1</u>	A2	A3

B	<u>B1</u>	<u>B2</u>	B3

D	<u>D1</u>	<u>D2</u>	D3

C bzw. R	<u>C1</u>	C2	C3	A1	B1	B2	D1	D2

In der letzten Tabelle existieren die folgenden Fremdschlüssel:

- A1 in A
- B1 und B2 in B
- D1 und D2 in D

(c) 2 Punkte

Erläutern Sie anhand eines konkreten Beispiels, was man unter einer Weak-Entity versteht.

Eine Weak-Entity ist eine Entity, deren "Elemente" nur dann existieren können,

wenn ein passendes "Element" in einer anderen Entity existiert. Zum Beispiel können Blutwerte eines Patienten nur dann existieren, wenn es auch den zugehörigen Patienten gibt.

Aufgabe 2 **(8 Punkte)**

Schreiben Sie ein **bash**-Skript `sum.sh`, das von der Standardeingabe Daten einliest. Jede Zeile ist durch den Trenner “:” in mehrere Spalten aufgeteilt. Ihr Skript soll als Ergebnis die Summe aller Zahlen in einer bestimmten Spalte ausgeben. Die Nummer der Spalte wird (bei 1 beginnend) auf der Kommandozeile angegeben. Wenn die angegebene Spalte nicht existiert oder keine Zahl enthält, dann soll sie ignoriert werden. Zahlen können negativ sein oder führende Nullen enthalten. Die Zeichen “+”, “,” oder “.” sind dagegen nicht zugelassen. Ihr Programm sollte bei einer Fehlbedienung eine freundliche usage-Meldung auf stderr ausgeben. Ob Sie Leerzeichen vor oder nach der Zahl unterstützen wollen, bleibt Ihnen überlassen.

```
#!/usr/bin/bash

if [ $# -ne 1 ]
then
        echo "usage: $0 number" >&2
        exit 1
fi

IDX="$1"
SUM=0
while read line
do
        OLDIFS="$IFS"
        IFS=":"
        set -- $line
        OLDIFS="$IFS"
        VAL=${!IDX}
        if echo $VAL | egrep "^ *-?[0-9][0-9]* *$" >/dev/null
        then
                SUM=$(($SUM + $VAL ))
        fi
done

echo "Summe: $SUM"
```

Aufgabe 3 **(5 Punkte)**

Welche Ausgabe produziert das folgende Programm? Bitte lesen Sie genau!

```
#!/usr/bin/bash

AAA=10
BBB=20

(
    BBB=30
    echo EINS: "$AAA $BBB"
)

echo "ZWEI:" "$AAA" "$BBB"

AAA=20

if [ AAA == BBB ]
then
    echo GLEICH
else
    echo UNGLEICH
fi

(
    BBB=50
    echo 'DREI: $AAA $BBB'
    BBB=60
    exit 1
)

echo VIER: $AAA $BBB
```

```
EINS: 10 30
ZWEI: 10 20
UNGLEICH
DREI: $AAA $BBB
VIER: 20 20
```

Aufgabe 4 **(8 Punkte)**

In der Datei `userdb` im aktuellen Verzeichnis stehen Einträge, die ungefähr wie folgt aussehen:

```
swg:geheim:130:200:Franz Schweiggert:/home/thales/swg:/bin/sh
ehrhardt:passwort:642:215:Christian Ehrhardt:/home/thales/ehrhardt:/bin/sh
```

Die einzelnen Spalten sind durch Doppelpunkte getrennt, in der ersten steht der login, in der fünften der zugehörige Name.
Die Ausgabe von `ls -l` sieht wie folgt aus:

```
drw-rw----    1 swg      sai          1516 Jan 27 15:58 dir
-rw-rw---t   12 ehrhardt staff          23 Jan 27 14:56 a file
```

Schreiben Sie ein **Perl**-Programm, das genau das selbe ausgibt wie `ls -l` – lediglich der login-Name soll durch den richtigen Namen ersetzt werden. Alles andere auf der Zeile soll unverändert bleiben. Insbesondere sollen alle ev. vorhandenen Leerzeichen und Tabs genau so erhalten bleiben wie sie `ls -l` ausgeben würde. Die Ausgabe sollte dann also wie folgt aussehen:

```
drw-rw----    1 Franz Schweiggert      sai          1516 Jan 27 15:58 dir
-rw-rw---t   12 Christian Ehrhardt staff          23 Jan 27 14:56 a file
```

```
#!/usr/local/bin/perl -w

use strict;
use IO::File;

my $fh = new IO::File ("<./userdb") || die "no passwd";
my %users = ();
while (<$fh>) {
        my @elms = split /:/;
        $users{$elms[0]} = $elms[4];
}
$fh->close ();

$fh = new IO::File ("ls -l|") || die "exec ls -l: $!";
while (<$fh>) {
        if (m/^(\S{10}\s+\d+\s+)(\S+)(\s.*$)/) {
                print "$1$users{$2}$3\n";
        } else {
                print;
        }
}
$fh->close ();
```

Aufgabe 5 **(6 Punkte)**

Die in Aufgabe 4 beschriebene Datei `userdb` enthält in der vierten Spalte die Nummer der Gruppe des Nutzers. Schreiben Sie ein Skript (wahlweise **Perl** oder **bash**), das angibt, wieviele Nutzer zu jeder einzelnen Gruppe gehören.

```
#!/usr/bin/bash

cut -d: -f4 ./userdb | sort | uniq -c
=================================================

#!/usr/local/bin/perl -w

use strict;
use IO::File;

my %hash = ();
my $fh = new IO::File ("<./userdb") || die "File not found";

while (<$fh>) {
        my (undef, undef, undef, $grp) = split /:/;
        if (exists $hash{$grp}) {
                $hash{$grp}++;
        } else {
                $hash{$grp} = 1;
        }
}

foreach (keys %hash) {
        printf ("%10d%10d\n", $_, $hash{$_});
}
```

Aufgabe 6 **(8 Punkte)**

In einer (erfundenen) Programmiersprache beginnen Kommentare immmer mit dem Zeichen "[" und enden mit dem Zeichen "]". Für unsere Zwecke besteht das gesamte Programm nur aus einer einzigen Zeile, die sich bereits in der Variablen `$_` befindet. Schreiben Sie ein Stück **Perl**-Code, der alle Kommentare aus dem Programm in `$_` entfernt, unter der Annahme, dass

(a) 4 Punkte
Kommentare geschachtelt werden können (wie in Oberon oder Modula-2)

```
while ( s/\[[^\[\]]*\]//) {}
```

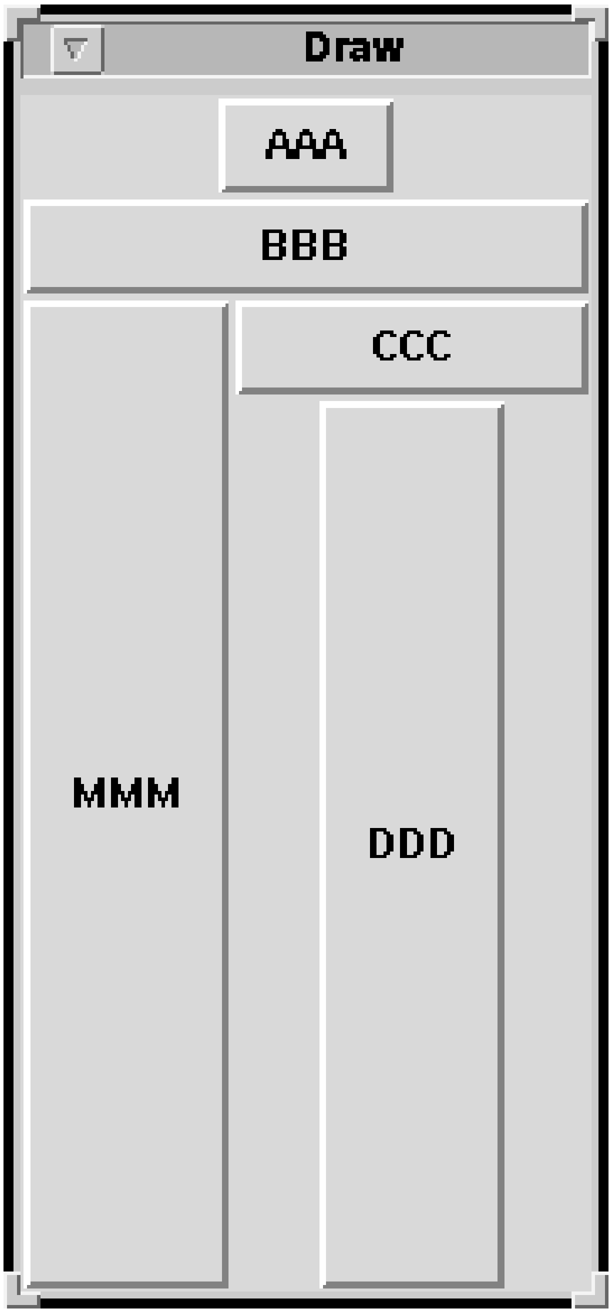
(b) 4 Punkte
Kommentare nicht geschachtelt werden können (wie in C)

```
s/\[[^\]]*\]//g
```

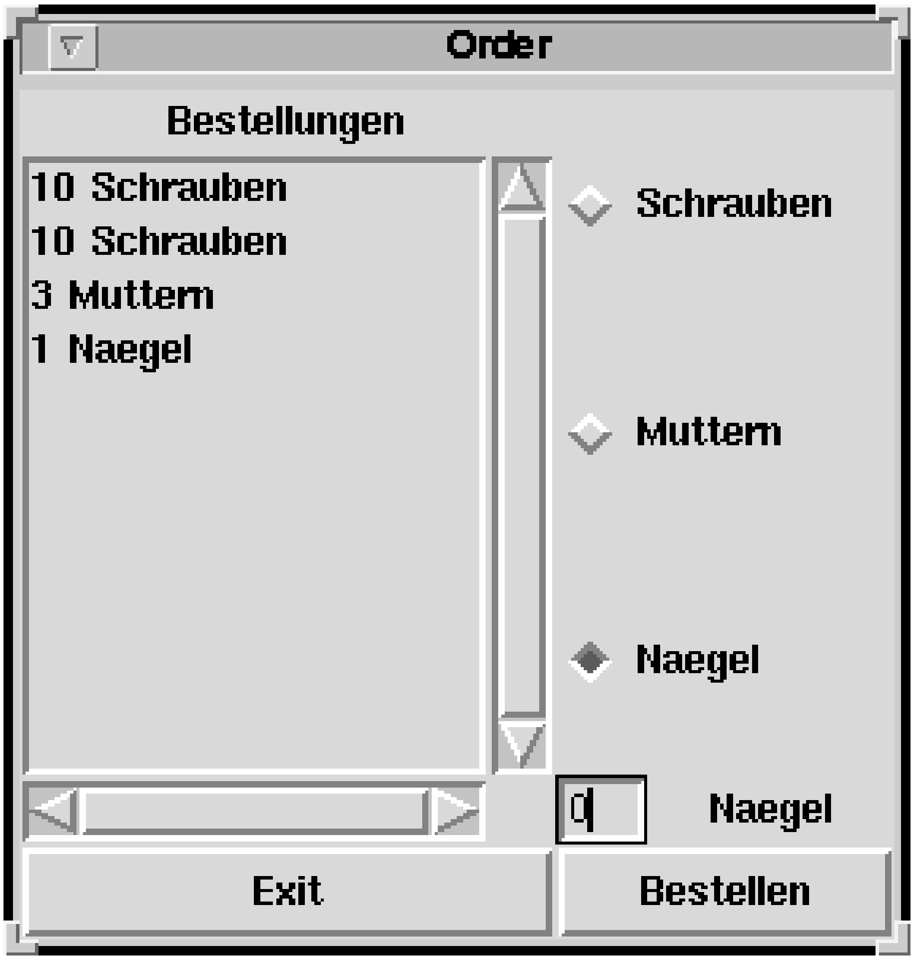
Aufgabe 7 **(8 Punkte)**

Gegeben sei das folgende **Perl**-Programm. Skizzieren Sie das Fenster, das von diesem Programm erzeugt wird.

```
#!/usr/local/bin/perl -w

use strict;
use Tk;

my $var = "XXX";
my $main = new MainWindow;

$main->Button (-text => "AAA")
        -> pack (-side => "top", -expand => 1);
$main->Button (-text => "BBB", -width => 20)
        -> pack (-side => "top");
$main->Button (-textvariable => \$var, -height => 20)
        ->pack (-side => "left", -fill => "both");
$main->Button (-text => "CCC")
        ->pack (-side => "top", -fill => "x");
$main->Button (-text => "DDD")
        ->pack (-side => "bottom", -fill => "y", -expand => 1);

$var = "MMM";

MainLoop;

$var = "OOO";
```


Draw
AAA
BBB
CCC
MMM
DDD

Aufgabe 8 **(12 Punkte)**



Schreiben Sie ein **Perl/Tk**-Programm, das das obige Fenster anzeigt. Das Programm soll folgendes Verhalten zeigen:

- Die Listbox ist zu Beginn leer.
- Beim Klicken auf den Exit-Button wird das Programm beendet.
- Beim Klicken auf einen der Radiobuttons ändert sich der Text neben dem Eingabefeld entsprechend dem Radiobutton und der Text im Eingabefeld selbst wird auf 0 gesetzt.
- Wenn beim Klicken auf den Bestellen-Button im Eingabefeld eine positive Zahl steht, wird die neue Bestellung zum bisherigen Inhalt der Listbox hinzugefügt. Ansonsten passiert nichts.

```
#!/usr/local/bin/perl -w

use strict;
use Tk;

my $current = "Schrauben";
my $count = "0";
my $main = new MainWindow;
my $left = $main->Frame () ->pack (-side => "left", -fill => "y");
my $list = $left->Scrolled ('Listbox', -scrollbars => "se",
        -label => "Bestellungen")
        ->pack (-side => "top", -fill => "both", -expand => 1);
my $button = $left->Button (-text => "Exit",
        -command => sub { exit (0); } )
        -> pack (-side => "top", -fill => "x");
foreach (qw {Schrauben Muttern Naegel}) {
        $main->Radiobutton (-text => "$_", -value => "$_",
                -variable => \$current,
                -command => sub { $count = 0; } )
                ->pack (-side => "top", -anchor => "w",
                        -fill => "y", -expand => 1);
}
my $frame2 = $main->Frame () ->pack (-side => "top", -fill => "x");
my $entry = $frame2->Entry (-width => 3, -textvariable => \$count)
        -> pack (-side => "left");
$frame2->Label (-textvariable => \$current) ->pack (-side => "left",
                -fill => "x", -expand => 1, -anchor => "w");
$main->Button (-text => "Bestellen", -command => sub {
                if (($count =~ m/^[0-9]+$/) && ($count > 0)) {
                        $list->insert ('end', "$count $current");
                        $count = 0;
                }
        }) -> pack (-side => "top", -fill => "x");

MainLoop;
```

Aufgabe 9 **(4 Punkte)**

Ergänzen Sie das folgende Programm so, dass das Text-Fenster beim Klicken in das Fenster die Anzahl der Zeichen in der Zeile ausgibt, in die geklickt wurde.

```
#!/usr/local/bin/perl -w

use strict;
use Tk;

my $main = new MainWindow;
my $edit = $main->Scrolled ("Text", -scrollbars => "se")->pack ();

# Hier kommt Ihr Part

$edit->bind ('<Button-1>', [ \&doit, Ev('@') ]);

sub doit
{
        my ($self, $pos) = @_;
        my $p = $edit->index ("$pos lineend");
        $p =~ s/^\d+\.//;
        print "$p\n";
}

MainLoop;
```