Klausur zu UNIX-basierte Implementierung kleiner Datenbanken (SS 2005) 4. Juli 2005

Bearbeitungszeit: 120 Minuten
NICHT MIT BLEISTIFT SCHREIBEN!

Name:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Nr	Max	Bewertung	
1	**16**	xxxxx	
(a)	8		xxxxx
(b)	8		xxxxx
2	**14**	xxxxx	
(a)	1		xxxxx
(b)	1		xxxxx
(c)	2		xxxxx
(d)	2		xxxxx
(e)	2		xxxxx
(f)	2		xxxxx
(g)	3		xxxxx
(h)	1		xxxxx

Nr	Max	Bewertung	
3	**8**	xxxxx	
4	**6**	xxxxx	
5	**10**	xxxxx	
6	**10**	xxxxx	
7	**8**	xxxxx	
8	**10**	xxxxx	
Summe	**82**		

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 bei Verwendung von `use strict;` und `use warnings;` ohne Warnungen laufen.

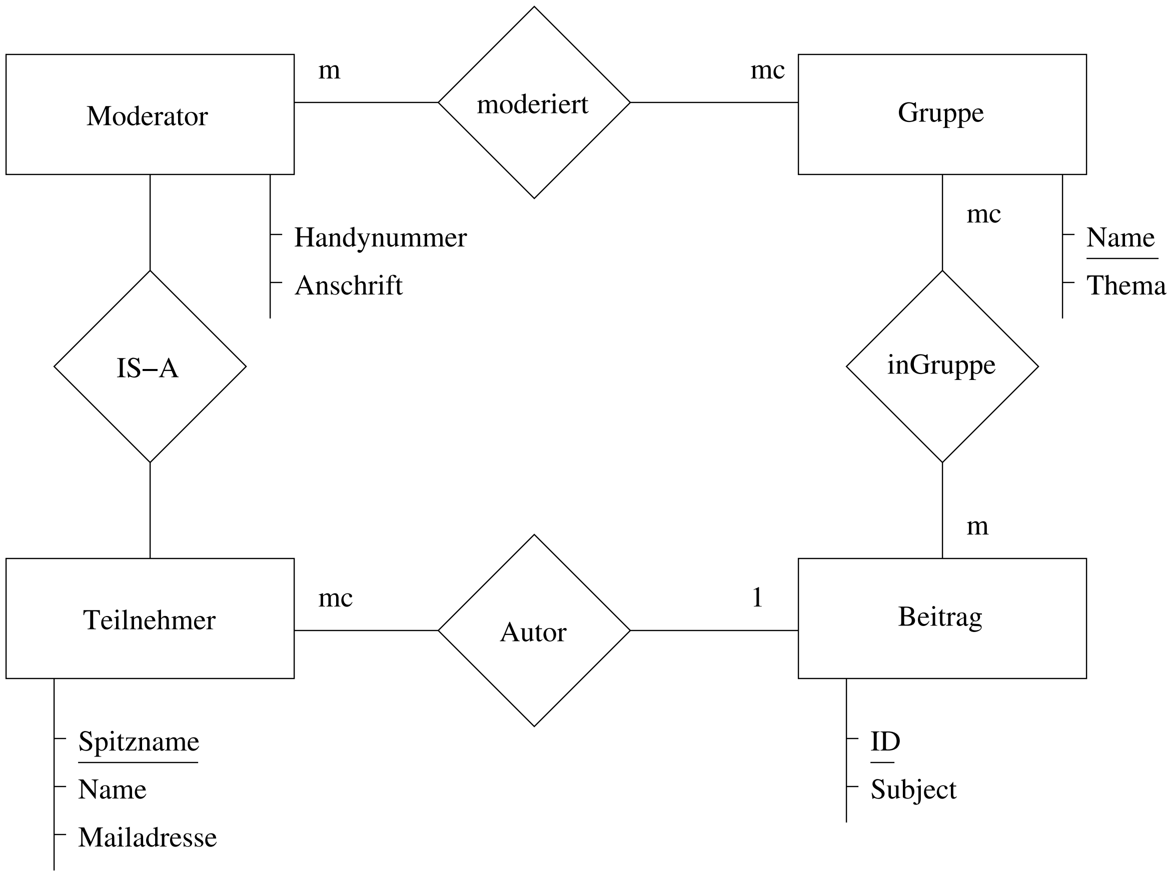
Viel Erfolg!

Aufgabe 1 **(16 Punkte)**

In einem Diskussionsforum gibt es verschiedene Teilnehmer, die Beiträge in verschiedenen Diskussiongruppen des Forums verfassen. Eine Diskussionsgruppe hat einen Namen und eine kurze Beschreibung des Diskussionsthemas. Jeder Diskussionsteilnehmer hat einen eindeutigen Spitznamen, einen Namen und eine Mailadresse. Außerdem gibt es Beiträge, ein Beitrag hat ein Subject und eine eindeutige Beitrags-ID. Jeder Beitrag kann in einer oder mehreren Diskussionsgruppen **gleichzeitig** stehen. Außerdem hat jeder Beitrag einen Autor, der ein Teilnehmer des Diskussionsforums sein muß. Ein Teilnehmer des Forums kann ein Moderator von einer oder mehreren Gruppen sein, in diesem Fall interessiert für Notfälle noch die Handynummer und die genaue Anschrift. Jede Diskussionsgruppe kann keinen, einen oder mehrere Moderatoren haben.

(a) 8 Punkte

Zeichnen Sie ein ER-Diagramm, das den oben beschriebenen Sachverhalt möglichst genau abbildet. Aus dem Diagramm soll hervorgehen, welche Attribute ein Objekt eindeutig identifizieren.



(b) 8 Punkte

Beschreiben Sie, welche Tabellen für die Realisierung dieser Beziehungen mit Hilfe einer relationalen Datenbank erforderlich sind. Primär- und Fremdschlüssel müssen eindeutig als solche gekennzeichnet sein.

Teilnehmer

- **Spizname** String, Primärschlüssel
- **Name** String
- **Mailadresse** String
- **IS-Moderator** "Y" oder "N"
- **Hanynummer** String, undefiniert falls kein Moderator
- **Anschrift** String, undefineiert falls kein Moderator

Beitrag

- **ID** String, Primärschlüssel
- **Subject** String
- **Autor** String, Fremdschlüssel in Teilnehmer

Gruppe

- **Name** String, Primärschlüssel
- **Thema** String

moderiert

- **Moderator** String, Primärschlüssel, Fremdschlüssel in Teilnehmer
- **Gruppe** String, Primärschlüssel, Fremdschlüssel in Gruppe

inGruppe

- **ID** String, Primärschlüssel, Fremdschlüssel in Beitrag
- **Gruppe** String, Primärschlüssel, Fremdschlüssel in Gruppe

Aufgabe 2 **(14 Punkte)**

Geben Sie **eine** Bash-Kommandozeile an, mit der genau all die Dateien im Verzeichnis `/tmp` gelöscht werden,

(a) 1 Punkt
deren Name auf `.txt` endet.

```
rm /tmp/*.txt
```

(b) 1 Punkt
deren Name mit `$` beginnt.

```
rm /tmp/\$*
```

(c) 2 Punkte
deren Name mit einem Punkt gefolgt von genau drei beliebigen Zeichen endet.

```
rm /tmp/*.???
```

(d) 2 Punkte
deren Name mindestens ein Leerzeichen enthält.

```
rm /tmp/*" "*
```

(e) 2 Punkte
deren Name das erste Zeichen aus der Umgebungsvariable AAA enthält.

```
rm /tmp/*"${AAA:0:1}"*
```

(f) 2 Punkte
deren Name den Text `$BBB=` gefolgt vom Inhalt von der Umgebungsvariablen `CCC` enthält.

```
rm /tmp/*'$BBB='"$CCC"*
```

(g) 3 Punkte
Wie muß die vorhergehende Kommandozeile abgeändert werden, wenn der Text `<undef>` statt dem Inhalt von `CCC` verwendet werden soll, falls die Umgebungsvariable CCC nicht definiert ist?

```
rm /tmp/*'$BBB='"${CCC:-<undef>}"*                oder
rm /tmp/*'$BBB='"${CCC:=<undef>}"*
```

(h) 1 Punkt

Welchen Wert hat die Umgebungsvariable `CCC` nach der von Ihnen angegebenen vorhergehenden Kommandozeile, wenn `CCC` vorher tatsächlich nicht definiert war?

```
rm /tmp/*'$BBB='"${CCC:-<undef>}"*    ==> CCC ist undefiniert
rm /tmp/*'$BBB='"${CCC:=<undef>}"*    ==> CCC="<undef>"
```

Aufgabe 3 **(8 Punkte)**

Gegeben sei ein Programm `logit`, das im Verzeichnis `/var/log` ein Protokoll seiner Aktivitäten (ein sog. Log-File) hinterläßt. Das Programm `logit` öffnet beim Start ein Log-File in `/var/log` und schließt das zugehörige File-Handle bis zum Programmende nicht mehr. Einzige Ausnahme: Wenn `logit` ein `HUP`-Signal erhält, dann wird das Log-File geschlossen und sofort wieder geöffnet. Die Prozeß-ID von `logit` steht in der Datei `/var/run/logit.pid`.
Da `logit` sehr viel Aktivität zeigt wächst die entsprechende Datei relativ schnell. Das kann zu Problemen führen, wenn das Programm auch relativ lange läuft. Ihre Aufgabe ist es, diesem Problem Abhilfe zu schaffen. Schreiben Sie dazu ein Bash-Skript `rotate`, das wie folgt funktioniert:

- Das Skript bekommt zwei Parameter, den Namen des Log-Files und eine kleine positive Zahl. Wir nehmen für die folgenden Erklärungen an, daß das durch den ersten Parameter angegebene Log-File `messages` heißt.
- Beim Aufruf soll das Skript dann `messages` in `messages.1` umbenennen. Ein ev. existierendes `messages.1` wird vorher in `messages.2` umbenannt usw.
- Es werden allerdings nur so viele Dateien behalten, wie der zweite Parameter angibt. War der zweite Parameter also z.B. `5`, dann wird `messages.4` noch in `messages.5` umbenannt, der alte Inhalt von `messages.5` geht verloren.
- `rotate` soll dafür sorgen, daß `logit` nach einer Rotation wieder in die Datei mit dem Namen `messages` schreibt.
- Es soll sichergestellt sein, daß Einträge in das Log-File, die während der Rotation stattfinden, nicht verloren gehen.

```
#!/usr/bin/bash

cd /var/log || exit 1

function usage
{
	echo "usage: $0 file count" >&2
	exit 2
}

if [ $# -ne 2 ]
then
	usage
fi

FILE=$1
MAX=$2
PID=$(cat /var/run/logit.pid)

COUNT=1

if [ -f ${FILE} ]
then
	while [ -f "${FILE}.${COUNT}" ]
	do
		COUNT=$(( $COUNT + 1 ))
	done
fi

if [ $COUNT -gt $MAX ]
then
	COUNT=$MAX
fi

while [ $COUNT -ge 2 ]
do
	NEXT=$(( $COUNT - 1))
	mv "${FILE}.${NEXT}" "${FILE}.${COUNT}" 2>/dev/null
	COUNT=$NEXT
done

mv "$FILE" "${FILE}.1" 2>/dev/null
kill -HUP $PID
```

Aufgabe 4 **(6 Punkte)**

In einer Datei stehen in gewohnter Weise Datensätze, ein Datensatz pro Zeile. Die einzelnen Datenfelder sind durch Doppelpunkte getrennt. Schreiben Sie ein Shell-Skript, das als ersten und einzigen Parameter den Namen der Datei erhält. Das Skript soll dann den Inhalt der fünften Spalte entfernen, falls in der dritten Spalte der gleichen Zeile genau der Text `\s` steht.

```
#!/usr/bin/bash

function usage
{
        echo "usage: $0 filename" >&2
        exit 2
}

if [ $# -ne 1 ] || ! [ -f "$1" ]
then
        usage
fi

<"$1" sed 's/^\([^:]*:[^:]*:\\s:[^:]*:\)[^:]*/\1/' >"$1._tmp$$"
mv -- "$1._tmp$$" "$1"
```

Aufgabe 5 **(10 Punkte)**

Die Ausgabe von `ps -ef` sieht etwa wie folgt aus:

```
     UID   PID  PPID  C    STIME TTY       TIME CMD
    root     0     0  0   Dec 24 ?         0:01 sched
ehrhardt  9214 13045  0   Mar 17 pts/166  0:13 /bin/bash -ovi
ehrhardt 18433 18426  0 11:23:55 pts/2     3:07 firefox
dnscache 18399   344  0   Jun 10 ?        26:26 /sbin/dnscache
ehrhardt 17925 17905  0 09:50:00 pts/2     0:00 bash
```

Schreiben Sie ein Perl-Skript, das für jeden Benutzer (Spalte `UID`) angibt, wieviel CPU-Zeit (Spalte `TIME`) seine noch laufenden Prozesse bisher verbraucht haben. Die Zeitangabe in der Spalte `TIME` besteht aus Minuten (vor dem Doppelpunkt) und Sekunden (nach dem Doppelpunkt). Die Zahl vor dem Doppelpunkt kann beliebig groß werden, die Zahl der Sekunden sollte natürlich nie größer als 59 sein. In der Spalte `STIME` steht entweder ein Datum oder eine Uhrzeit, die Spalte `TTY` enthält ein Fragezeichen oder den Text `pts/` gefolgt von einer Zahl.
Für jeden Nutzer soll ausgegeben werden:

- Der Login der Nutzers,
- die **gesamte** von allen Prozessen des Users verbrauchte CPU-Zeit in der Form wie sie in der Spalte `TIME` steht,
- die Kommandozeile (Spalte `CMD`), des Kommandos, das den größten Anteil an der verbrauchten CPU-Zeit des Nutzers hat und
- die von **diesem** Kommando verbrauchte CPU-Zeit.

Das alles soll **absteigend** nach der gesamten CPU-Zeit der Nutzer sortiert werden.

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

use strict;
use warnings;
use IO::File;

my $fh = new IO::File ('ps -ef | grep -v \'<defunct>$\' |') || die "$!";

<$fh>;


my %h = ();

while (<$fh>) {
        chomp;
        s/^\s*//;
/^(\w+)\s+\d+\s+\d+\s+\d+\s+[\w:]+(\s+\d+)?\s+\D\S*\s+(\d+):(\d+)\s+(.*)$/;
        my ($uid, $min, $sec, $cmd) = ($1, $3, $4, $5);
        my $t = 60 * $min + $sec;
        if (!defined ($h{$uid})) {
                $h{$uid} = {
                        sum => $t,
                        max => $t,
                        time => "$min:$sec",
                        cmd => $cmd
                }
        } else {
                $h{$uid}->{sum} += $t;
                if ($t > $h{$uid}->{max}) {
                        $h{$uid}->{max} = $t;
                        $h{$uid}->{time} = "$min:$sec";
                        $h{$uid}->{cmd} = $cmd;
                }
        }
}
close ($fh);

for (sort {$h{$b}->{sum} <=> $h{$a}->{sum}} keys %h) {
        my $s = $h{$_}->{sum} % 60;
        $s = "0$s" if (length ($s < 2));
        my $m = ($h{$_}->{sum} - $s) / 60;
        print "$_ ($m:$s) $h{$_}->{cmd} ($h{$_}->{time})\n";
}
```

Aufgabe 6 **(10 Punkte)**

In einer Liste stehen die Namen von berühmten und weniger berühmten Persönlichkeiten und möglicherweise Ihre Geburts- und Sterbedaten. Es ist bekannt, daß in jeder Zeile der Eingabe die Daten zu genau einer Person stehen, Geburts- und Sterbedatum stehen dabei (falls vorhanden) am Ende der Zeile. Der Name einer Person besteht für die Zwecke dieser Aufgabe nur aus Klein- und Großbuchstaben sowie Whitespace. Etwas komplizierter ist es beim Datum. Hier gelten die folgenden Regeln:

- Der Monat wird immer als Zahl angegeben.
- Geburts- und Sterbedatum werden durch ein Minuszeichen getrennt. Dieses Minuszeichen darf in einer korrekten Zeile niemals fehlen.
- Fehlt das Geburtsdatum, gilt es als unbekannt.
- Fehlt das Sterbedatum, gilt die betreffende Person als noch am Leben.
- Ein vollständiges Datum besteht aus Tag, Monat und Jahr, wobei negative Jahreszahlen "vor Christi Geburt" bedeuten. Als Trenner dient entweder ein Punkt (".", europäische Schreibweise) oder ein Schrägstrich ("/", amerikanische Schreibweise). Ein einzelnes Datum enthält nur einen der beiden Trenner.
- Bei der amerikanischen Schreibweise sind Tag und Monat vertauscht.
- Wenn das genaue Datum nicht bekannt ist, dann kann in beiden Schreibweisen der Tag oder der Tag und der Monat entfallen.

Schreiben Sie ein Perl-Skript, das überprüft, ob eine Zeile in diesem Sinne korrekt ist, d.h. ob sie genau aus einem Namen gefolgt von Geburts- und Sterbedatum wie oben beschrieben besteht. Korrekte Zeilen sollen so umformatiert werden, daß sowohl Geburts- als auch Sterbedatum in europäischer Notation ausgegeben werden. Jeder fehlende Teil eines Datums (Tag, Monat, Jahr) soll druch `??` ersetzt werden, so daß jedes Datum in der Ausgabe aus genau drei Komponenten besteht.

Beispiel für eine Eingabe, die nur aus korrekten Zeilen besteht:

```
Caesar Julius --44
Julius Caesar amerikanisch -3/15/-44
Julius Caesar europaeisch -15.3.-44
Christian Ehrhardt 28.12.1975-
Johann Wolfgang von Goethe 28.8.1749-3/22/1832
Johanna Christiane Sophie Vulpius 06/1765-6.1816
Jesus Christus -4-33
Mohammed 570-8.6.632
```

Die zugehörige Ausgabe sollte dann so aussehen:

```
Caesar Julius  ??.??.??-??.??.-44
Julius Caesar amerikanisch  ??.??.??-15.3.-44
Julius Caesar europaeisch  ??.??.??-15.3.-44
Christian Ehrhardt  28.12.1975-??.??.??
Johann Wolfgang von Goethe  28.8.1749-22.3.1832
Johanna Christiane Sophie Vulpius  ??.06.1765-??.6.1816
Jesus Christus  ??.??.-4-??.??.33
Mohammed  ??.??.570-8.6.632
```

Hinweis: Das Minuszeichen, das die beiden Datumsangaben voneinander trennt, ist das einzige von möglicherweise mehreren Minuszeichen in einer Zeile, das direkt auf eine Ziffer folgen kann.

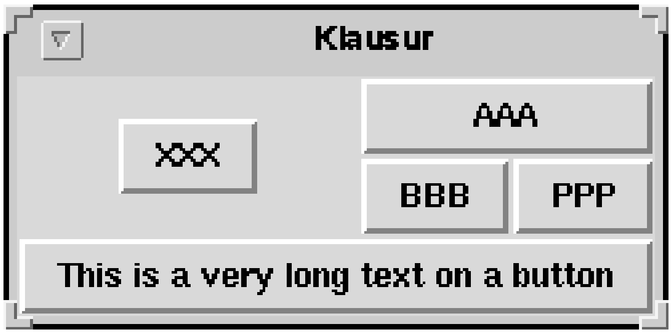
```
#!usr/local/bin/perl

use strict;
use warnings;

while (<>) {
        chomp;
        next unless m/^([A-Za-z\s]+)([\d\-\/\.]+)$/;
        my ($name, $dates) = ($1, $2);
        next unless $dates =~ m/\-/;
        my ($d1, $d2);
        if ($dates =~ m/^(.*\d)\-(.*)/) {
                $d1 = $1;
                $d2 = $2;
        } else {
                next unless $dates =~ m/^\-(.*)$/;
                $d1 = "";
                $d2 = $1;
        }
        my $date1 = check_date ($d1);
        my $date2 = check_date ($d2);
        next unless defined ($date1) and defined ($date2);
        print "$name ";
        print join (".", @{$date1});
        print "-";
        print join (".", @{$date2});
        print "\n";
}
```

```
sub check_date
{
	my ($date) = @_;
	if ($date eq "") {
		return ["??", "??", "??"];
	}
	if ($date =~ m/^(\d+\/){0,2}(\-?\d+)$/) {
		my @ret = split /\//, $date;
		if (scalar @ret == 3) {
			my $tmp = $ret[1];
			$ret[1] = $ret[0];
			$ret[0] = $tmp;
		}
		while (scalar @ret < 3) {
			unshift (@ret, "??");
		}
		return \@ret;
	}
	if ($date =~ m/^(\d+\.){0,2}(\-?\d+)$/) {
		my @ret = split /\./, $date;
		while (scalar @ret < 3) {
			unshift (@ret, "??");
		}
		return \@ret;
	}
	return undef;
}
```

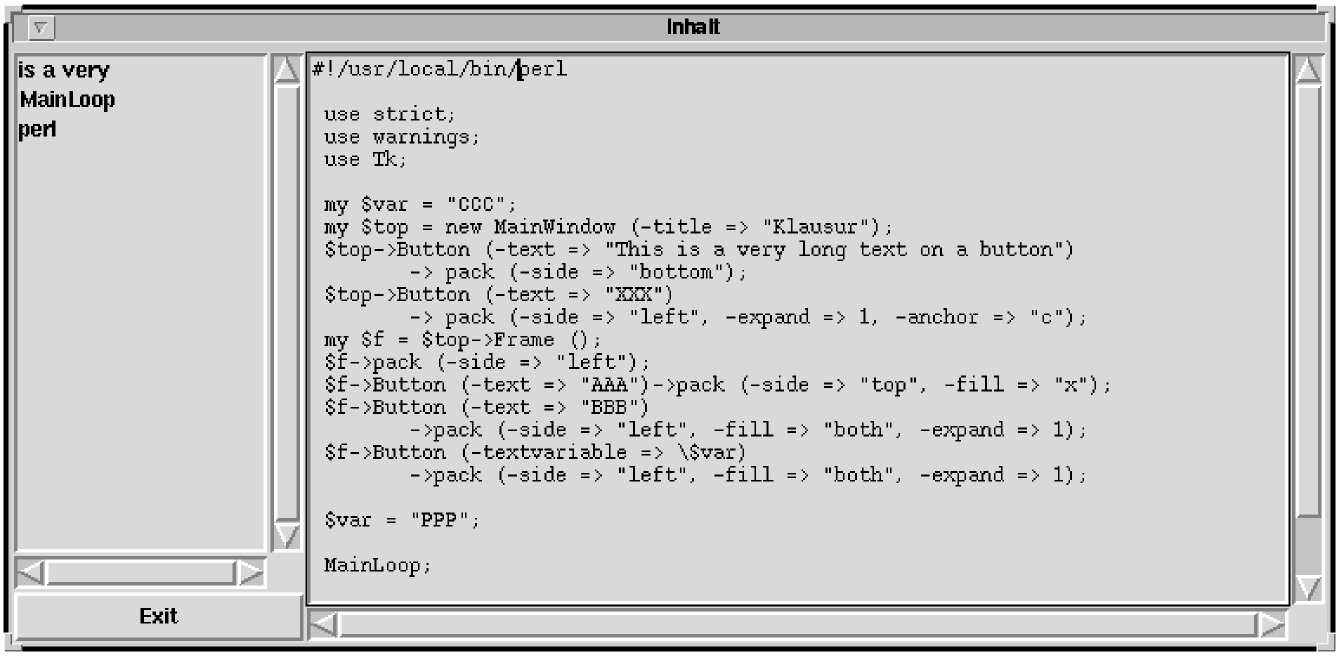
Aufgabe 7 **(8 Punkte)**

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

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

use strict;
use warnings;
use Tk;

my $var = "CCC";
my $top = new MainWindow (-title => "Klausur");
$top->Button (-text => "This is a very long text on a button")
        -> pack (-side => "bottom");
$top->Button (-text => "XXX")
        -> pack (-side => "left", -expand => 1, -anchor => "c");
my $f = $top->Frame ();
$f->pack (-side => "left");
$f->Button (-text => "AAA")->pack (-side => "top", -fill => "x");
$f->Button (-text => "BBB")
        ->pack (-side => "left", -fill => "both", -expand => 1);
$f->Button (-textvariable => \$var)
        ->pack (-side => "left", -fill => "both", -expand => 1);

$var = "PPP";

MainLoop;

$var = "QQQ";
```



Aufgabe 8 **(10 Punkte)**



Schreiben Sie ein PerlTK-Programm, das dieses Fenster erzeugt und folgende Funktion hat:

- Das Programm soll den Namen einer Datei als ersten Parameter übergeben bekommen. Diese Datei wird (nicht editierbar) im Textfenster rechts dargestellt.
- Wird im Textfenster die Taste F8 gedrückt, dann wird der im Moment markierte Text in die Listbox links übernommen.
- Ist ein mehrzeiliger Text markiert oder gar kein Text markiert, dann passiert nichts. Die Listbox ist beim Programmstart leer.
- Wird ein Eintrag in der Listbox ausgewählt, dann wird umgekehrt die zugehörige Stelle im Text rechts angesprungen (sichtbar gemacht).
- Es kann also auf diese Weise eine Art Bookmark-Liste erzeugt werden.
- Ein Klick auf den “Exit”-Button beendet das Programm.

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

use strict;
use warnings;
use IO::File;
use Tk::ROText;
use Tk;

die "usage: $0 file\n" unless @ARGV == 1;

my $fh = new IO::File ($ARGV[0]) || die "$!";
my @lines = <$fh>;
close ($fh);

my $main = new MainWindow ();

my $txt = $main->Scrolled ('ROText', -scrollbars => "se");
$txt->pack (-side => "right", -fill => "both", -expand => 1);

my $list = $main->Scrolled ('Listbox', -scrollbars=> "se");
$list->pack (-side => "top", -fill => "both", -expand => 1);

$main->Button (-text => "Exit", -command => sub { exit (0); })
        ->pack (-fill => "both");
$txt->insert ("1.1", "@lines");

my %pos = ();

$txt->bind ('<KeyPress-F8>', [ \&insert ] );
$list->bind ('<ButtonPress-1>', sub {
                my @sel = $list->curselection ();
                return if (scalar @sel != 1);
                my $entry = $list->get (@sel);
                return unless defined ($pos{$entry});
                $txt->see($pos{$entry});
        });


MainLoop;

sub insert
{
        my @range = $txt->tagRanges ('sel');
        return unless @range == 2;
        my $l1 = $range[0];
```

```
        $l1 =~ s/\..*//;
        my $l2 = $range[1];
        $l2 =~ s/\..*//;
        return unless $l1 == $l2;
        my $word = $txt->get (@range);
        if (defined ($pos{$word})) {
                $pos{$word} = $range[0];
        } else {
                $pos{$word} = $range[0];
                $list->delete (0, "end");
                $list->insert (0, sort { lc ($a) cmp lc ($b) } keys %pos);
        }
}
```