Musterlösung zu Blatt 9:

1. Rechnen sie folgende Zahlen in Binärzahlen, Oktalzahlen und Hexadezimalzahlen um:
a) 15, b) 22, c) 256, d) 512, e) 1024, f) 2048
(freiwillige Zusatzaufgabe: Gibt es eine Möglichkeit vcn einer Kodierungen in eine andere mit PERL umzurechnen? Schreiben Sie ein PERL Programm, das von Dezimal auf Oktal und Hexadezimal umrechnet)

Dezimal	**Binär**	**Oktal**	**Hexadezimal**
15	1111	17	F
22	10110	26	16
256	100000000	400	100
512	1000000000	1000	200
1024	10000000000	2000	400
2048	100000000000	4000	800

```
#!/usr/bin/perl
# Autor: Nicola Greth
# Rechnet Dezimalzahlen in Binaer-, Oktal- und Hexadezimalzahlen um
use strict;
{
  my ($dezimal, $eingabe, @oktal, $oktalergebnis, @hexadezimal,
$hexadezimalergebnis, @binaer, $binaerergebnis);
  print"Bitte geben Sie Ihre Dezimalzahl ein:>>>\n";
  chomp($eingabe=<>);

  #Umrechnung in Binaer
  $dezimal = $eingabe;
  my $i = 0;
  while($dezimal > 0) {
    $binaer[$i] = $dezimal%2;
    #speichert in @oktal den Rest von $dezimal : 2
    $dezimal = int($dezimal/2);
    #Dezimalzahl wird durch 2 geteilt, int (damit der Rest abgeschnitten wird)
    $i++;
  }

  @binaer = reverse(@binaer);
  $binaerergebnis = join("",@binaer);
   print"Ihre Dezimalzahl $eingabe lautet $binaerergebnis in Binaerzahlen.\n";
```

```
  #Umrechnung in Oktal
  $dezimal = $eingabe;
  $i=0;
  while($dezimal > 0) {
    $oktal[$i] = $dezimal%8;
    #speichert in @oktal den Rest von $dezimal : 8
    $dezimal = int($dezimal/8);
    #Dezimalzahl wird durch 8 geteilt, int (damit der Rest abgeschnitten wird)
    $i++;
  }

  @oktal = reverse(@oktal);
  $oktalergebnis = join("",@oktal);
  print"Ihre Dezimalzahl $eingabe lautet $oktalergebnis in Oktalzahlen.\n";

  #Umrechnung in Hexadezimal
  $dezimal = $eingabe;
  $i = 0;
  while($dezimal > 0) {
    $hexadezimal[$i] = $dezimal%16;
    #speichert in @oktal den Rest von $dezimal : 16
    if ($hexadezimal[$i] == 10) {
      $hexadezimal[$i] = "A";
    }
    if ($hexadezimal[$i] == 11) {
      $hexadezimal[$i] = "B";
    }
    if ($hexadezimal[$i] == 12) {
      $hexadezimal[$i] = "C";
    }
    if ($hexadezimal[$i] == 13) {
      $hexadezimal[$i] = "D";
    }
    if ($hexadezimal[$i] == 14) {
      $hexadezimal[$i] = "E";
    }
    if ($hexadezimal[$i] == 15) {
      $hexadezimal[$i] = "F";
    }
    $dezimal = int($dezimal/16);
    #Dezimalzahl wird durch 16 geteilt, int (damit der Rest abgeschnitten wird)
    $i++;
  }


  @hexadezimal = reverse(@hexadezimal);
  $hexadezimalergebnis = join("",@hexadezimal);
  print"Ihre Dezimalzahl $eingabe lautet $hexadezimalergebnis in
Hexadezimalzahlen.\n";
}
```

2. Wie sind folgende Zeichenketten in UTF-8 - und in ISO-LATIN-1 Kodierung abgespeichert? (Tipp: Speichern Sie die Wörter in einer utf8/isolatin1 Datei und betrachten Sie, analog zur Vorlesung, den octal Dump der Datei mit dem unix-Befehl od)
a) Zeichenkette : 'Weiß'
b) Zeichenkette : "ÄäÜüÖö"

in IsoLatin:
0000000 062527 157551 142012 156344 153374 000366
0000013

in UTF-8:
0000000 062527 141551 005237 102303 122303 116303 136303 113303
0000020 133303
0000022

In UTF-8 gibt es mehr Bytes, da Sonderzeichen und Umlaute als Mehrbytecharacter dargestellt werden.

3. Legen Sie eine neue Textdatei "inp.txt" an und schreiben Sie in die Datei das Wort "über". Öffnen Sie die Datei 'inp.txt' erneut im Editor und speichern Sie die Datei als ISO-LATIN Datei : iso.txt und als UTF-8 Datei utf8.txt ab.
a) Wieviele Bytes stehen in de Datei inp.txt, iso.txt und der Datei utf8.txt ?
b) Welche Dateien sind identisch?
c) Bestätigen sie das auch mit dem UNIX Befehl 'diff'. Wie lauten die UNIX Befehle? Wie lautet der OUTPUT von der UNIX Befehle?

diff -qs inp.txt iso.txt
diff -qs iso.txt utf8.txt
diff -qs inp.txt utf8.txt

→ welche Dateien identisch sind hängt davon ab in welcher Codierung inp.txt standardmäßig gespeichert wurde

od inp.txt → 4 Bytes/ 5 Bytes (hängt von standardmäßiger Codierung ab)
od iso.txt → 4 Bytes
od utf8.txt → 5 Bytes

4. Zwei Wörter sind Anagramme, wenn in ihnen die gleichen Buchstaben in beliebiger Reihenfolge vorkommen (oma<->mao).
Schreiben Sie zwei PERL Programme, die testen ob zwei Wörter Anagramme sind.

1. Programm: read_2_anas.perl (beide Wörter einlesen)

```
#!/usr/bin/perl
# Autor: Nicola Greth
# Programm: Liest zwei Woerter ein und testet, ob es sich um Zwei Anagramme
handelt

use strict;
use warnings;
use locale;
{
  my ($word1, $word2, @buchst1, @buchst2);
  my $anagramme = "true";

  print "Bitte geben Sie das erste Wort ein:>>>\n";
  chomp($word1 = <>);
  $word1 = lc($word1);
  print "Bitte geben Sie das zweite Wort ein:>>>\n";
  chomp($word2 = <>);
  $word2 = lc($word2);

  @buchst1 = split(//,$word1);
  @buchst2 = split(//,$word2);
  @buchst1 = sort(@buchst1);
  @buchst2 = sort(@buchst2);

  if(length($word1) != length($word2)) {
    $anagramme = "false";
    print "Ihre Woerter sind keine Anagramme, da die Laenge nicht
uebereinstimmt!\n";
  }

  else {
    for(my $i=0;$i<length($word1);$i++) {
      if($buchst1[$i] ne $buchst2[$i]) {
        $anagramme = "false";
      }
    }

    if ($anagramme eq "true") {
      print "$word1 und $word2 sind Anagramme!\n";
    }
    else {
      print "$word1 und $word2 sind keine Anagramme!\n";
    }
  }
}
```

2. Programm: read_1_ana.perl und Aufgabe 5:

```
#!/usr/bin/perl
# Autor: Nicola Greth
# Programm: Liest zwei Woerter ein und testet, ob es sich um Zwei Anagramme
handelt

use strict;
use warnings;
use locale;
{
  my ($word_intern, $word2, @buchst1, @buchst2);
  my $anagramme = "true";

  $word_intern = "oma";
  print "Suchen Sie das Anagramm:>>>\n";
  # Aufgabe 4
  #print "Suchen Sie ein Anagramm zum wort $word_intern:>>>\n";
  # Aufgabe 5
  chomp($word2 = <>);
  $word2 = lc($word2);

  @buchst1 = split(//,$word_intern);
  @buchst2 = split(//,$word2);
  @buchst1 = sort(@buchst1);
  @buchst2 = sort(@buchst2);

  if(length($word_intern) != length($word2)) {
    $anagramme = "false";
    print "Ihre Woerter sind keine Anagramme, da die Laenge nicht
uebereinstimmt!\n";
  }

  else {
    for(my $i=0;$i<length($word_intern);$i++) {
      if($buchst1[$i] ne $buchst2[$i]) {
        $anagramme = "false";
      }
    }

    if ($anagramme eq "true") {
      print "$word_intern und $word2 sind Anagramme!\n";
    }
    else {
      print "$word_intern und $word2 sind keine Anagramme!\n";
    }
  }
}
```

6./7./8.Funktioniert Ihr Programm auch bei Umlauten? Bauen Sie nun in Ihr Programm die Anweisung: use utf8; ein. Was passiert jetzt bei Anagrammen, die Umlauten, bzw. keine Umlaute enthalten? Rufen sie jetzt ihr PERL Programm mit der Option -C auf. Was passiert jetzt?

Funktioniert nicht mit Umlauten.
Mit use utf8; und perl-C funktioniert es auch mit Umlauten.
Hinweis: achtet darauf dass ihr die Datei als utf8-Datei speichert (→ Speichern unter)

9. Schreiben Sie mit Hilfe von einfachen print Anweisungen, die Sie in Wiederholungsschleifen einbauen, ein PERL Programm, das folgende Ausgabe auf der Console erzeugt.
Bemerkung: print soll nur einzelne Buchstaben wie Spaces, Slashes, Punkte, Tilde, Sterne, Eckige Klammern, Underline, Kommas usw. ausdrucken, alles andere soll mit Wiederholungsschleifen realisiert werden.

```
#!/usr/bin/perl
# Autor: Nicola Greth
# Programm: malt Weihnachtsbaum

use strict;
{
  my $blank=" ";
  my $star="*";
  my $forwardslash="/";
  my $backslash="\\";
  my $dot=".";
  my $comma=",";
  my $roof="^";
  my $bracketLeft="[";
  my $bracketRight="]";
  my $underscore="_";
  my $newline="\n";
  my $lines;
  my $width;

  print "Bitte geben Sie die Hoehe des Baumes ein (4-...): ";
  chomp($lines = <>);
  $width = ($lines/2);

  for(my $l=0;$l<$lines-1;$l++) {
  #Leerzeichen am Anfang der Zeile
    for(my $w=0;$w<$width;$w++) {
      print $blank;
    }
    #fuer die erste Zeile
    if($l==0) {
      print $star;
      $width--;
```

```
    }
    else {
    #fuer gerade Zeilen
      if($l%2==0) {
        print $forwardslash;
          for(my $i=0;$i<$l-1;$i++) {
            if($i%2==0) {
              print $comma;
            }
            else {
              print $dot;
            }
          }
        print $backslash;
        $width--;
       }
      #fuer ungerade Zeilen
      elsif($l%2!=0) {
        print $forwardslash;
        for(my $i=0;$i<$l;$i++) {
          if($i%2==0) {
            print $dot;
          }
          else {
            print $comma;
          }
        }
        print $backslash;
      }
    }
  print $newline;
  }

  for(my $i=0;$i<$lines/2-1;$i++) {
    print $roof;
  }
  print $bracketLeft;
  print $underscore;
  print $bracketRight;

  for(my $i=0;$i<($lines/2)-1;$i++) {
    print $roof;
  }
  print $newline;
}
```

oder:

Musterlösung der Gruppe Sophia Antonin:

```
#!/usr/bin/perl

use strict;
{
	my $i;
	my $z;
	for ($i=6; $i>0; $i--) {
		if ($i == 6) {
			print " "x$i."*\n";
		} else {
			$z=1;
			while ($z < 3) {
				print " "x$i."/";
				if ($z == 1) {
					print "."."x(5-$i)."\\\n";
				} else {
					print "."."x(5-$i)."\\\n";
				}
				$z++;
			}
		}
	}
	print "^"x(5)."[_]"."^"x(5)."\n";
}
```