Lösungen der AuD Beispielprüfungsaufgaben (http://www.schimmeier.de/tutor/aud.html):

aud_aufg_01:

Ausgabe:

```
E= H=   2H=   2H=   4H=   3H=  12 12
```

Zu beachten ist bei dieser Aufgabe, daß **zuerst** `E=` ausgegeben und danach erst angefangen wird zu berechnen:

```
    writeln ('E= ', f(x):3);
```

Desweiteren ist der Seiteneffekt zu beachten (bei Pascal ←—):

```
    h := f(n - 1) * f(n - 3);
```

Es wird also **zuerst** `f(n - 3)` und dann `f(n - 1)` berechnet.

aud_aufg_03:

Programmablaufplan:

```
function fac (n : integer) : integer;  // besser: function fac(n : word) : word;
var
  i, p : integer;  // oder word
begin
    p := 1;
    i := 1;
  while i <= n do begin
    p := p * i;
  end;
    fac := p;
end;
```

Struktogramm:

```
function fac (n : integer) : integer;  // besser: function fac (n : word) : word;
var
  i, p : integer;  // oder word
begin
    p := 1;
    i := n;
  if i = 0 then else repeat  // besser: if i <> 0 then repeat
    p := p * i;
    dec (i);  // i := i - 1;
  until i = 0;
    fac := p;
end;
```

aud_aufg_04:

Der Suchlauf entstammt nicht einem binären Suchbaum, weil die 40 nicht am selben Teilast hängen würde wie die 80 (40 < 50 und 80 > 50).

Andere Erklärung: die 120 würde nicht am selben Teilast wie die 70 hängen (70 < 80 und 120 > 80).

aud_aufg_05 (Spaghetti-Code ist nichts gegen diese Liste)**:**

a)

```
Type
  PRec = ^TRec;
  TRec = Record
           Z1   : PRec;
           Z2   : PRec;
           Info : Char;
         End;

Var
  Anfang : PRec;
```

b)

```
    New (Anfang);
  With Anfang^ Do Begin
    Info := 'A';   // Anfang^.Info := 'A';
    New (Z1);          Z1^.Info := 'B';
    New (Z1^.Z1);      Z1^.Z1^.Info := 'C';
    New (Z1^.Z1^.Z1); Z1^.Z1^.Z1^.Info := 'D';
                       Z1^.Z1^.Z1^.Z1   := Nil;
    New (Z2);          Z2 := Z1^.Z1;
    New (Z1^.Z2);      Z1^.Z2 := Z2^.Z1;
    New (Z2^.Z2);      Z2^.Z2 := Z1^.Z1;
    New (Z1^.Z2^.Z2); Z1^.Z2^.Z2 := Anfang;
  End;
```

c)

Ausgabe:

```
ADDC
```

aud_aufg_06:

Ausgabe:

```
0
5
```

Das letzte Ergebnis (Zeile 33, `Lc`) wird nie ausgegeben werden, da die Liste `Lc` eine Endlosschleife produziert:

```
While z <> Nil Do
```

Das letzte Element der Liste zeigt auf das erste – `z` wird also niemals `Nil`.

aud_aufg_07:

3)

HR 1 2 4 5 7 3 6
NR 4 7 5 2 6 3 1
SR 4 2 7 5 1 6 3

4)

Anhand der SR sieht man, daß der Baum kein binärer Suchbaum ist, da sie SR einen binären Suchbaum **sortiert** ausgibt.