Anlage zu Aufgabe 4

```
#include <stdio.h>
#include <stdlib.h>

#define BB_B 4           /* maximale Anzahl der Soehne pro Knoten */

typedef int schluesseltyp;
typedef int rest_typ;

typedef struct datensatz
{
  schluesseltyp schluessel;
  rest_typ uebrige_daten;
}
datensatz;

typedef enum {bb_wurzel, bb_inner, bb_blatt} bb_knoten_typ;

typedef struct bb_knoten                /* Knoten eines B-Baumes */
{
  bb_knoten_typ typ;                          /* Typ des Knotens */
  union         /* uebriger Inhalt des Knotens, abhaengig vom Typ */
  {
    struct              /* Struktur fuer Wurzel und innere Knoten */
    {
      int anz_soehne;                        /* Anzahl der Soehne */
      struct bb_knoten *sohn[BB_B];      /* Zeiger auf die Soehne */
      schluesseltyp schluessel[BB_B];   /* zugehoerige Schluessel */
    } wi;
    /* ACHTUNG: Aenderung gegenueber Vorlesung: */
    struct                          /* in den Blattknoten: Daten */
    {
      datensatz daten;
      struct bb_knoten *nachf; /* NEU */
    } blatt;
  } inhalt;
} bb_knoten;

typedef bb_knoten *b_baum;

/*****************************************************************/

void bb_ausgabe(b_baum b)
{
  static int tiefe = -1;
  int i;

  if (!b)
    {
      printf("Baum leer\n");
      return;
    }
  tiefe++;
  if (b->typ == bb_blatt)
    {
      for (i = 0; i < tiefe; i++)
        printf("         ");
      printf("| %i |", b->inhalt.blatt.daten.schluessel);
      /* Nachfolger ausgeben */
      if (b->inhalt.blatt.nachf == NULL)
        printf("   nachf:  NULL\n");
      else
        printf("   nachf:  %i\n",
               b->inhalt.blatt.nachf->inhalt.blatt.daten.schluessel);
    }
  else
    {
      for (i = b->inhalt.wi.anz_soehne - 1; i >= 0; i--)
        bb_ausgabe(b->inhalt.wi.sohn[i]);
      for (i = 0; i < tiefe; i++)
        printf("       ");
      printf("( ");
      for (i = 0; i < b->inhalt.wi.anz_soehne; i++)
```

```
        printf("%i ", b->inhalt.wi.schluessel[i]);
      for (i = b->inhalt.wi.anz_soehne; i < BB_B; i++)
        printf("- ");
      printf(")\n");
    }
  tiefe--;
}

/***************************************************************/

void nachf_erzeugen1(b_baum b, bb_knoten **letzter)
{
  /* HIER ERGAENZEN */

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void nachf_erzeugen(b_baum b)
{
  bb_knoten *letzter = 0;
  nachf_erzeugen1(b, &letzter);
}

/***************************************************************/

void print_blaetter(b_baum b)
{
  bb_knoten *bkn;

  printf("Blätter: ");
  if (!b)
    /* bei leerem Baum ist nichts zu tun */
    return;
  /* erstes Blatt suchen */
  bkn = b;
  while (bkn->typ != bb_blatt)
    bkn = bkn->inhalt.wi.sohn[0];
  /* dann: Blaetter durchlaufen */
  while (bkn != NULL)
    {
      printf("%i ", bkn->inhalt.blatt.daten.schluessel);
      bkn = bkn->inhalt.blatt.nachf;
    }
  printf("\n");
}

/***************************************************************/

int main(void)
{
  int i, n;
  b_baum b = NULL;
  datensatz d;

  printf("Anzahl der Elemente: ");
  scanf("%i", &n);
  for (i = 0; i < n; i++)
    {
      printf("%i-tes Element: ", i);
      scanf("%i", &d.schluessel);
      bb_einfuegen(d, &b);
    }

  nachf_erzeugen(b);

  bb_ausgabe(b);

  print_blaetter(b);

  return 0;
}
```