zu Aufgabe 4:

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

#define MAX_N 50
#define MAX_CODE MAX_N

typedef double werte[MAX_N];          /* Gewichte der Blattknoten */

typedef int code[MAX_CODE];        /* code fuer ein Wort          */

typedef int bb_datentyp; /* im Binaerbaum zu speichernden Daten,
                            hier:  int (Nr. des Wortes)         */

typedef struct bb_knoten
{
  bb_datentyp bb_daten;             /* im Knoten abgelegte Daten */
  struct bb_knoten *bb_lsohn;           /* Zeiger auf linken und */
  struct bb_knoten *bb_rsohn;                   /*  rechten Sohn */
}
bb_knoten;
/*****************************************************************/

void codes_ausgeben( bb_knoten * b, code c, char *woerter[MAX_N],
                     int tiefe)

     /* gibt die dem Binaerbaum entsprechenden Codes aus */
{
  int i;

  if (b != NULL)
    {
      if (( b->bb_lsohn == NULL ) && ( b->bb_rsohn == NULL ))
        /* Blattknoten erreicht; Code ausgeben */
        {
          printf("\nCode ");
          for (i = 0; i <= tiefe; i++)
            printf("%d", c[i]);
          printf(" fuer das Wort %s", woerter[b->bb_daten] );
        }
      else
        {
          c[ tiefe + 1 ] = 0;
          codes_ausgeben( b->bb_lsohn, c, woerter, tiefe + 1 );
          c[ tiefe + 1 ] = 1;
          codes_ausgeben( b->bb_rsohn, c, woerter, tiefe + 1 );
        }
    }
}

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

void codierungsbaum( int n, werte m, bb_knoten **pb)
{

  /* pb ist Zeiger auf Binaerbaum; Binaerbaum ist Zeiger
     auf bb_knoten                                          */
  double unendlich = 1.0e37;            /* ziemlich grosse Zahl */

  Queue q;           /* Warteschlange fuer die noch zu unter-
                        suchenden und die bereits untersuchten (im
                        naechsten Schritt wieder zu bearbeitenden)
                        Teilbaeume                                */
  Eintrag_typ min1, min2;
  /* die beiden Teilbaeume mit dem kleinsten
     bzw. zweitkleinsten Gewicht              */
  int i, j;
  Eintrag_typ tmp;

  q_create( &q );
  for (i=0;i<n;i++)
    {
      /* neuen Knoten fuer Binaerbaum erzeugen */
```

```
      if ((tmp.baum = (bb_knoten*)malloc(sizeof(bb_knoten)))
          == NULL)
        {
          printf("Kein Speicher mehr frei!\n");
          exit(EXIT_FAILURE);
        }
      tmp.baum->bb_daten = i;      /* Nummer des Worts eintragen */
      tmp.baum->bb_lsohn = NULL;
      tmp.baum->bb_rsohn = NULL;
      tmp.gewicht = m[i];     /* Gewicht des Teilbaums eintragen */
      enqueue( tmp, &q );
    }

  /* (n - 1)-mal zwei Teilbaeume zusammenfassen */
  for (i = 0; i <n-1; i++)
    {
      min1.gewicht = unendlich;
      min2.gewicht = unendlich;
      for (j = 0; j< n - i; j++)  /* leichtesten und 2.leichtesten */
        {                           /* Baum suchen                    */
          tmp = dequeue( &q );
          if (tmp.gewicht < min2.gewicht)
            {
              if (min2.gewicht != unendlich)
                enqueue( min2, &q );
              if (tmp.gewicht < min1.gewicht)
                {
                  min2 = min1;
                  min1 = tmp;
                }
              else
                min2 = tmp;
            }
          else
            enqueue( tmp, &q );
        }

      if ((tmp.baum = (bb_knoten*)malloc(sizeof(bb_knoten)))
          == NULL)
        {
          printf("Kein Speicher mehr frei!\n");
          exit(EXIT_FAILURE);
        }
      tmp.baum->bb_lsohn = min1.baum;
      tmp.baum->bb_rsohn = min2.baum;
      tmp.gewicht = min1.gewicht + min2.gewicht;
      enqueue( tmp, &q );
    }
  tmp = dequeue( &q );
  *pb = tmp.baum;
}

/* NEU ***********************************************************/

void print_decoded(bb_knoten * b, code c, int codelen, char *woerter[MAX_N])
{
  /* HIER ERGAENZEN */

}

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

int main(void)
{
  int i;
  bb_knoten *b;                 /* der zu bestimmende Binaerbaum */
  code c;                                /* der zugehoerige code */
  char string_tmp[20];         /* Hilfsfeld um strings einzulesen */
  char *woerter[MAX_N];                 /* zu codierende Woerter */
  werte m;
  int n;                                    /* Anzahl der Woerter */
  int codelen;


```

```
  printf("Anzahl der Woerter eingeben\n");
  scanf("%d",&n);
  for (i = 0; i< n; i++)
    {
      getchar();                              /* new line ueberlesen */
      printf("\n%dtes Wort eingeben: ",i);
      fgets(string_tmp,19,stdin);
      woerter[i] = (char*)malloc(strlen(string_tmp));
      strcpy(woerter[i],string_tmp);
      printf("und dessen Haeufigkeit ");
      scanf("%lf", &m[i] );
    }

  codierungsbaum( n, m, &b );/* bestimme optimalen Codierungsbaum */
  codes_ausgeben( b, c, woerter, -1 );

  /** NEU **/
  printf("Code eingeben:\n");
  printf("Länge des Codes: ");
  scanf("%i", &codelen);
  if (n > 0)
    {
      for (i = 0; i < codelen; i++)
        {
          printf("%i.tes Bit: ", i);
          scanf("%i", &c[i]);
        }

      print_decoded(b, c, codelen, woerter);
    }

  return 0;
}
```