```c
/* ==================================================================
/* = Uebung 8, Aufgabe 1                                       = */
/* = Verkette Listen                                           = */
/* ==================================================================

#include <stdio.h>
#include <stdlib.h>
/* Auf manchen Systemen muss man noch spezielle includes fuer
   die Funktion 'time' benutzen. Siehe 'man time' */



/* STRUKTUREN ======================================================
struct element
{
  int wert;
  struct element *vorgaenger;
  struct element *nachfolger;
};



/* FUNKTIONEN ======================================================
/* Aufgabe 1 (a) ------------------------------------------ */

/* ein neues Element erzeugen und initialisieren */
struct element *neues_element()                                    neues_element
{
  struct element *neues;

  /* Speicher fuer das neue Elemente allokieren */
  neues = (struct element *)malloc(sizeof(struct element));
  /* pruefen, ob es geklappt hat */
  if (neues == NULL)
  {
    printf("Kann kein neues Element anlegen.\n");
    exit(-1);
  }

  /* neues Element mit Werten belegen */
  neues->wert = 0;
  neues->vorgaenger = NULL;
  neues->nachfolger = NULL;

  return neues;                /* neues Element zurueckgeben */
}
```



```c
/* eine Menge neuer Elemente erzeugen */
struct element *neue_kette(int anzahl)                             neue_kette
{
  struct element *neues;
  struct element *altes=NULL;
  int i;

  /* 'anzahl' viele Elemente erzeugen */
  for(i=0; i<anzahl; i++)
  {
    neues = neues_element();          /* neues Element erzeugen */
    /* jetzt wird das neue Element vorne angehaengt (weil's leichter ist) */
    /* den Nachfolger setzten (falls es einen gab) */
    neues->nachfolger = altes;
    /* Wenn es einen Nachfolger gab, dann dessen Vorgaenger setzen */
    if (neues->nachfolger != NULL)
      neues->nachfolger->vorgaenger = neues;
    /* jetzt ist das neue das alte Element und wir sind fertig */
    altes = neues;
  }

  /* das letzte neu erzeugte Element ist das erste und wird zurueckgegeben */
  return neues;
}



/* eine Liste von Elementen loeschen (von hinten nach vorne rekursiv) */
void loesche_kette(struct element *start)                          loesche_kette
{
  if (start->nachfolger != NULL)
  {
    /* wenn es einen Nachfolger gibt, loesche diesen zuerst */
    loesche_kette(start->nachfolger);
    free(start);  /* dann loesche dich selbst */
  }
  else
  {
    /* wenn es keinen Nachfolger gibt, ist das Element das letzte */
    /* und wird geloescht */
    free(start);
  }
}



/* Aufgabe 1 (b) ------------------------------------------ */
```

random_init

```
/* eine Liste von Elementen mit Zufallswerten fuellen */
void random_init(struct element *start)
{
  /* solange noch Elemente in der Liste sind */
  if (start != NULL)
  {
    start->wert = rand();          /* Wert erzeugen */
    random_init(start->nachfolger); /* Rest der Liste initialisieren */
  }
}
```

drucke_kette

```
/* Aufgabe 1 (c) ------------------------------------------------ */
void drucke_kette(struct element *start)
{
  static int i=0;

  /* solange noch Elemente in der Liste sind */
  if (start != NULL)
  {
    if (i%4 < 3) printf("(%3d: %7d)  ", i, start->wert);
    else         printf("(%3d: %7d)\n", i, start->wert);
    i++;
    drucke_kette(start->nachfolger); /* Rest der Liste drucken */
  }
  else i=0;
}
```

suche_minimum

```
/* Aufgabe 1 (d) ------------------------------------------------ */
/* eine Hilfsfunktion, die das Minimum sucht */
struct element *suche_minimum(struct element *start)
{
  struct element *min;          /* Minimum */

  if (start->nachfolger != NULL)
  {
    /* Suche das Minimum im Rest der Kette */
    min = suche_minimum(start->nachfolger);
    if (min->wert < start->wert)  /* Wenn das Minimum kleiner ist als der */
      return min;                 /* momentane Wert, diesen zurueck, ... */
    else
      return start;               /* ... sonst den momentanen Wert */
  }
  else
    return start;                 /* Wenn es nur noch ein Element gibt, ... */
```



```
}
```

sort_kette

```
/* sortiere die Elemente einer Liste rekursiv */
struct element *sort_kette(struct element *start)
{
  struct element *min;          /* Minimum */
  struct element *naechstes;

  /* wenn noch mindestens zwei Elemente in der Liste sind */
  if (start->nachfolger != NULL)
  {
    min = suche_minimum(start); /* suche das Minimum in der Liste */

    /* wenn start nicht das Minimum ist, ... */
    if (min != start)
    {
      /* ... dann das Minimum aus der Liste entfernen */
      if (min->vorgaenger != NULL) min->vorgaenger->nachfolger = min->nachfolger;
      if (min->nachfolger != NULL) min->nachfolger->vorgaenger = min->vorgaenger;
      naechstes = start;  /* der Rest der Liste beginnt mit start */
    }
    else
    {
      /* den Rest der Liste nach vorne abschliessen */
      start->nachfolger->vorgaenger = NULL;
      /* wenn start das Minimum ist dann ist der Rest der Liste alles hinter start */
      naechstes = start->nachfolger;
    }

    /* Jetzt wird die sortierte Liste wieder zusammengebaut */
    min->vorgaenger = NULL;  /* die Liste nach vorne abschliessen */
    min->nachfolger = sort_kette(naechstes);  /* den sortierten Rest anhaengen */
    min->nachfolger->vorgaenger = min;  /* den vorgaenger vom Rest der Liste setzen */
  }
  else
  {
    /* wenn nur noch ein element in der Liste ist */
    start->vorgaenger = NULL;  /* Liste nach vorne abschliessen */
    return start;  /* start ist das Minimum, da es sonst keinen Wert gibt */
  }

  /* Das Minimum dieser Rekursionsstufe zurueckgeben */
  return min;
}
```

```
/* ==================================================================
/* ------------------------------------------------------------- */
/* HAUPTPROGRAMM                                                 */
/* ------------------------------------------------------------- */
/* ==================================================================
int main()                                                          main
{
  struct element *start;
  int anzahl;


  printf("Anzahl der Elemente eingeben: ");
  scanf("%d", &anzahl);
  printf("Erzeuge %d neue Elemente.\n", anzahl);
  start = neue_kette(anzahl);
  drucke_kette(start);

  printf("\nInitialisiere Liste random...\n");
  random_init(start);
  drucke_kette(start);

  printf("\nSortiere Liste...\n");
  start = sort_kette(start);
  drucke_kette(start);

  printf("\nLoesche Liste...\n");

  loesche_kette(start);
  printf("\n - ENDE -\n");

  return 0;
}
```