Aufgabe 1: Spezifizieren Sie das C-Programm PrintProduktListe inklusive einer geeigneten Fehlerbehandlung. PrintProduktListe erhält über die Kommandozeile einen Dateinamen übergeben und gibt, falls die Datei existiert, die Werte aller in der Datei gespeicherten Datensätze zeilenweise auf den Standard-Output aus. Dabei darf davon ausgegangen werden, daß - im Falle der Existenz der Datei - in dieser eine Folge von Datensätzen der Struktur:

```
struct ProduktDatensatz {
   long ProduktNummer
   char ProduktName[31];
   float ProduktPreis;
};
```

gespeichert ist.

Aufgabe 2: Spezifizieren Sie die C-Funktion GetNextProductData in zwei Varianten. Die Funktion erhält als Parameter einen File-Pointer zu eine Datei, die Datensätze der Struktur

```
struct ProduktDatensatz {
   long ProduktNummer
   char ProduktName[31];
   float ProduktPreis;
};
```

enthält; die Funktion liest den nächsten Datensatz aus dieser Datei und gibt ihn an den Aufrufer:

- in der ersten Variante per Funktionsergebnis zurück und
- in der zweiten Variante per Paramter zurück.

Aufgabe 3: Spezifizieren Sie die C-Funktion CopyList. Die Funktion erhält mittels Parameter den Listenanfang einer verketetten Liste übergeben und fertigt eine Kopie der Liste an; die Adresse des ersten Listenelements der Kopie soll per Parameter an den Aufrufer übergeben werden.

Bei der Spezifikation der Funktion ist von folgendem Aufbau eines Listenelementes und den Datentypen seiner Komponenten auszugehen:

```
typedef struct Listenelement {
                  int Zahlenwert;
                  struct Listenelement *Naechstes;
               } LISTENELEMENT;
```

Aufgabe 1:

```
int main (argc, argv[])
{ fopen ...        struct ProduktDatensatz Datensatz;
  while ( !feof (fp))
  {
    if (fread ((void*)&Datensatz, sizeof (struct ProduktDatensatz),
                1, fp) == 1)
    { printf ("%ld, %s, %f \n", Datensatz.ProduktNummer,
              Datensatz.ProduktName, Datensatz.ProduktPreis);
    }
    else break;
  }
  ...
```

Aufgabe 2:

Variante 1: void Variante 2:

```
[struct ProduktDatensatz*] GetNextProductData (FILE *fp)
                                , struct ProduktDatensatz
                                  ** NextData
{
  [struct ProduktDatensatz *NextData;]
  *NextData = (struct ProduktDatensatz*) malloc (sizeof (struct
                                  ProduktDatensatz));
  if (*NextData == NULL)
    [return NULL;] return;
  if (fread ((void*)*NextData, sizeof (struct ProduktDatensatz),
                                  1, fp) == 1)
    [return NextData;] return;
  else      [free (NextData);]  free (*NextData);
    [return 0;][*NextData = NULL; return;]
}
```

Aufgabe 3:

```
int     CopyListe (struct Listenelement *Liste, struct Listenelement **Listen
                                                                    Kopie)
{   if (Liste == NULL)   { *ListenKopie = NULL
                           return OK; }
    while (Liste != NULL)
    {
        if (Anfuegen (ListenKopie, (*Liste).Zahlenwert) == HEAPERROR)
            return HEAPERROR;
        Liste = (*Liste).Naechstes;
    }
    return OK;
}
```

Knotenanzahl mittels static-Variable:

ohne Variable

```
int KnotenAnzahl (struct BaumKnoten *Wurzel)
{
    if (Wurzel == NULL)
        return 0;
    else
        return 1+ KnotenAnzahl((*Wurzel).Linker)+ KnotenAnzahl((*Wurzel).Rechte
}
```

```
int KnotenAnzahl (struct Baumknoten *Wurzel)
{
    static int Knoten =0;  static int RekursionsTiefe=0;
    if (Wurzel != NULL)
    {
        Knoten ++; RekursionsTiefe ++
        KnotenAnzahl ((*Wurzel).Linker);
        KnotenAnzahl ((*Wurzel).Rechter;
        RekursionsTiefe -- ; }
```

```
if (RekursionsTiefe == 0)
{
    int Ergebnis;
    Ergebnis = Knoten; Knoten = 0;
    return Ergebnis;
}
else
    return Knoten;
}
```

feof, ferror, fread, fscanf, fopen, fclose, fseek, fprintf, fwrite, fgetc, fputc, fgets, fputs

Dateibearbeitungsfunktionen