2. Praktische Übung in "Programmieren": Lagerverwaltung

Die Lagerverwaltung einer Drogerie wird rechnergestützt durchgeführt. Die Daten aller Artikel sind in einer Datei (Lagerdatei) auf einem Massenspeicher (Harddisk) abgelegt.
Für jeden Artikel enthält die Datei einen Eintrag, der durch den folgenden Structure-Typ dargestellt wird:

```
struct artikel_t
{ int artnr;                /* Artikelnummer                      */
  char artname[MAXCHAR];    /* Beschreibung (Name) des Artikels   */
  float preis;              /* Preis für ein Stück                */
  int bestand;              /* vorhandene Stückzahl               */
  int min;                  /* Mindestanzahl für die Bevorratung  */
};
```

Ihr Mitarbeiter hatte die Aufgabe, ein Modul "**lagdat.c**" zu realisieren, mit dem zu dem Massenspeicher zugegriffen werden kann. Dieses Modul enthält als Schnittstelle für den Dateizugriff die drei folgenden Funktionen :

1. **int openLager(void);**
   Aktion: Öffnen der Lagerdatei und Positionierung auf den ersten Artikel falls Datei noch nicht geöffnet ist
   Übergabeparameter: keine
   Rückgabewert: `0` Öffnen erfolgreich
   `1` kein erneutes Öffnen der Datei, da Datei bereits geöffnet ist
   `-1` im Fehlerfall (Lagerdatei kann nicht geöffnet werden !)

2. **int readNext(struct artikel_t *);**
   Aktion: Liest die Daten des nächsten Artikels in die Variable, die durch den als Parameter übergebenen Zeiger referiert wird. Zusätzlich wird auf den folgenden Artikel positioniert. Beim Leseversuch am Dateiende bzw im Fehlerfall ist der Inhalt der referierten Variablen undefiniert.
   Übergabeparameter: Zeiger auf eine Variable vom Typ "`struct artikel_t`", in der die gelesenen Daten des nächsten Artikels abgelegt werden.
   Rückgabewert: `0` falls Lesen erfolgreich
   `-1` beim Leseversuch am Dateiende oder im Fehlerfall (Datei ist nicht geöffnet).

3. **int closeLager(void);**
   Aktion: Schließen der Lagerdatei
   Übergabeparameter: keine
   Rückgabewert: `0` falls fehlerfrei
   `-1` im Fehlerfall (Schließen nicht erfolgreich)

Das Modul "**lagdat.c**" steht Ihnen zur Verfügung. Die Headerdatei "**lagdat.h**" enthält die Funktionsprototypen der drei Funktionen sowie die Definition des Typs "`struct artikel_t`".

Die ebenfalls bereitgestellte Datei "**lager.dat**" ist die aktuelle Lagerdatei. Sie enthält die Artikeldaten in maschineninterner Binärdarstellung.

a) **Formulieren** Sie die Funktion **main()** eines C-Programms, in der
- die **Gesamtzahl** unterschiedlicher Artikel und
- der **Gesamtwert** aller Artikel, die sich im Lager befinden, ermittelt
- und diese Werte anschließend in geeigneter Form auf dem Bildschirm (Standardausgabe) ausgegeben werden!

Falls sich die **Lagerdatei nicht öffnen** lässt, ist eine entprechende **Meldung** in die Standardausgabe auszugeben.

Rückgabewert von `main()` : `EXIT_SUCCESS (== 0)` bei Erfolg
`EXIT_FAILURE (== 1)` bei Misserfolg (Lagerdatei. lässt sich nicht öffnen)

Anmerkung : `EXIT_SUCCESS` und `EXIT_FAILURE` sind in `<stdlib.h>` definierte Konstante

weiter Seite 2

Die **nachfolgend beschriebenen Funktionen** **`bestellMenge()`** und **`bestellUebersicht()`** sind in einer **eigenen Quelldatei** mit dem Namen "**`lagfunc.c`**" zusammenzufassen.
Ihre **Funktionsdeklarationen** sollen in einer **Headerdatei** mit dem Namen "**`lagfunc.h`**" enthalten sein.

Vor ihrer jeweiligen **Implementierung** ist für jede Funktion ein **Struktogramm** zu erstellen.

- Mittels einer Funktion **`bestellMenge()`** sollen die Artikel ermittelt werden, deren **Mindestanzahl unterschritten** ist und die deshalb **nachbestellt** werden müssen.
  Die Funktion trägt die entsprechenden Artikel in ein Array (Elementtyp `struct artikel_t`) ein, dessen Anfangsadresse und Größe (Elemente-Anzahl) ihr als Parameter übergeben werden.
  Ist das übergebene Array zu klein, um alle nachzubestellenden Artikel aufzunehmen, wird die Anzahl der Artikel, die nicht mehr in das Array aufgenommen werden können, über einen geeigneten dritten Funktionsparameter zurückgegeben.
  Rückgabewert der Funktion : **Anzahl** der in das Array tatsächlich **eingetragenen Artikel**
  bzw `-1` im Fehlerfall (Lagerdatei lässt sich nicht öffnen)

b) **Realisieren** Sie die Funktion **`bestellMenge()`**.

Ergänzen Sie `main()` um
- die **Definition**
  - eines geeigneten **Arrays** zur Aufnahme der nachzubestellenden Artikel (Nachbestell-Array), die **Array-Größe** ist dabei durch eine **symbolische Konstante** festzulegen,
  - einer Variable zur Aufnahme des Rückgabewerts von `bestellMenge()` und
  - einer Variablen zur Aufnahme der Anzahl der Artikel, die im Nachbestell-Array keinen Platz mehr finden
- sowie den **Aufruf** der Funktion **`bestellMenge()`** und der anschließenden **Überprüfung**, ob alle nachzubestellenden Artikel im Nachbestell-Array aufgenommen werden konnten. Geben Sie **gegebenenfalls** eine entsprechende **Meldung** in die Standardausgabe aus.

- Die Funktion **`bestellUebersicht()`** hat die Aufgabe, eine Bestell-Liste für die mittels der Funktion `bestellMenge()` (s. b)) ermittelten Artikel zu erstellen. Bestellt werden sollen jeweils so viel Exemplare eines Artikels, dass der Vorrat auf das `ANZ`−fache seiner Mindestanzahl aufgefüllt wird. (`ANZ` symbolische Konstante mit dem Wert `5`). Die Liste soll in folgender Form in die Standardausgabe ausgegeben werden :

```
Lfd.Nr.  Art.Nr.  Artikel            Bestellanzahl Einzelpreis Gesamtpreis
-------------------------------------------------------------------------
   1       008    Deodorant               102          7.00       714.00
   2       006    Chrisam                  92          6.20       570.40

 ...       ...    ..................      ...       ........    .........
  11       038    Ohropax                  62          6.80       421.60
-------------------------------------------------------------------------
                                                    Summe:      3821.50
```

Rückgabewert der Funktion : keiner

c) **Überlegen** Sie, welche **Parameter** die Funktion **`bestellUebersicht()`** besitzen muss.
Realisieren Sie die Funktion **`bestellUebersicht()`** und rufen Sie sie in `main()` auf.

d) **Überlegen** Sie, wie Sie durch **Änderung** des **Rückgabetyps** von **`bestellMenge()`** auf den 3. Parameter dieser Funktion verzichten können. **Definieren** Sie einen geeigneten **Rückgabetyp**.
Formulieren Sie hiermit eine entsprechende **Funktionsdeklaration** von **`bestellMenge()`.**

e) **Option für alle, die sich noch fit fühlen** : lehrreich, wünschenswert, interessant,
Die Funktion **`bestellSort()`** hat die Aufgabe das Array der nachzubestellenden Artikel nach **ansteigenden Artikelnummern** zu **sortieren**.
Ein anschließender Aufruf von `bestellUebersicht()` erzeugt daraufhin eine **geordnete Bestell-Liste**.
Ergänzen Sie Ihr **Programm** um diese Funktionalität.