Aufgabe 1

Es soll ein Datenmodul erstellt werden, das einen sequentiellen Vektor und Operationen darauf realisiert. Programme, die dieses Modul verwenden (Anwendungsprogramme), sollen nicht direkt, sondern nur durch den Aufruf von Schnittstellenfunktionen (Operationen) auf diesen Vektor zugreifen können. Alle Datenobjekte und Funktionen des Moduls sollen in einer Quelldatei enthalten sein.
Die für die Verwendung dieses Moduls notwendigen Vereinbarungen sind dem Anwendungsprogramm in Form einer Header-Datei zur Verfügung zu stellen.
Bei der hier zu implementierenden Version des Moduls soll der Vektor mit der festen Länge 16 definiert werden. Im einzelnen sind die folgenden Schnittstellenfunktionen zu implementieren:

`unsigned` **`size(`** `void` **`);`**
: gibt die Anzahl der in dem Vektor abgespeicherten Elemente (= **aktuelle** Länge des Vektors) zurück.

`unsigned` **`capacity(`** `void` **`);`**
: liefert die Anzahl der Elemente, für die Speicherplatz bereitgestellt wurde. (Der von `capacity()` gelieferte Wert ist nie kleiner als `size()`.)

`void` **`append(`** `short int elem` **`);`**
: fügt ein neues Element mit dem Wert `elem` hinter dem letzten Element des Vektors ein.

`short int` **`getelem(`** `void` **`);`**
: der erste Aufruf nach Beginn der Programmausführung und jeder einem Aufruf von `reread` folgende Aufruf liefert das erste Element des Vektors. Andernfalls wird das Vektorelement zurückgegeben, das dem zuletzt durch `getelem` geholten Element unmittelbar folgt. (Der Resultatwert ist undefiniert, falls kein weiteres Vektorelement existiert.)

`void` **`reread(`** `void` **`);`**
: legt fest, dass der nächste Aufruf von `getelem` das erste Element des Vektors zurückgeben soll.

`int` **`findelem(`** `short int elem, int lastfoundelem` **`);`**
: sucht in dem Vektor nach dem nächsten Vorkommen des Wertes `elem`. Der Aufruf `findelem(elem,` **`START_SEARCH`**`)` gibt immer den Index des **ersten** Vorkommens zurück bzw. einen Wert **`NOT_FOUND`**, falls der gesuchte Wert überhaupt nicht im Vektor enthalten ist.
Durch eine Folge von Aufrufen, bei denen jeweils der Resultatwert des vorherigen Aufrufs beim nächsten Aufruf an `lastfoundelem` übergeben wird, können alle Vorkommen des gesuchten Wertes in dem Vektor ermittelt werden.
Der Aufruf `findelem(elem,` **`NOT_FOUND`**`)` soll **`NOT_FOUND`** als Resultat liefern.

`void` **`clear(`** `void` **`);`**
: löscht alle Elemente im Vektor.

Zu Beginn der Programmausführung soll der Vektor noch keine Elemente enthalten.

Zum Testen Ihres Vektor-Moduls können Sie das Programm `Public/vekmod1_main.c` verwenden.

Aufgabe 2

Zum Testen des Vektor-Moduls ist ein Programm zu schreiben, das den sequentiellen Vektor des Moduls mit Zufallszahlen im Bereich von 0 bis 9 besetzt. Anschließend soll für jede Zahl 0 bis 9 mittels der Funktion `findelem` bestimmt werden, wo diese im Vektor vorkommt.

Geben Sie den Vektor mit den Zufallszahlen und anschließend für jede der Zahlen 0 bis 9 ihre Positionen im Vektor (durch Angabe der Indizes) aus.

zusätzlich benötigte Biblotheksfunktionen zur Erzeugung von Pseudo-Zufallszahlen:

```
Library:                          Syntax:
 Standard C Library (libc.a)       #include <stdlib.h>
                                   int rand(void);
                                   void srand(unsigned int seed);
```

Die Funktion `rand` liefert eine ganzzahlige Pseudo-Zufallszahl im Bereich von 0 bis **`RAND_MAX`**; dieser Wert ist mindestens 32767. Die Funktion `srand` dient dazu, vor dem ersten Aufruf von `rand`, einen Startwert für die Folge von Pseudo-Zufallszahlen zu setzen. `rand` liefert für einen bestimmten Ausgangswert `seed` immer die gleiche Folge von Zufallszahlen (Deshalb werden diese als *Pseudo*-Zufallszahlen bezeichnet.). Wird `srand` nicht aufgerufen, dann ist der Ausgangswert 1.

Damit nicht jedesmal die gleichen Zufallszahlen erzeugt werden, empfiehlt es sich, den Startwert in Abhängigkeit von der Uhrzeit zu wählen. Z.B. könnte der Wert der folgenden Funktion als Startwert benutzt werden:

```
Library:                          Syntax:
 Standard C Library (libc.a)       #include <time.h>
                                   time_t time(time_t *tp);
```

`time` liefert die aktuelle Kalenderzeit oder `-1`, wenn diese nicht zur Verfügung steht. `time_t` ist ein arithmetischer Typ. Wenn der Zeiger `tp` von **`NULL`** verschieden ist, wird der Resultatwert auch in dem Objekt, auf das `tp` zeigt, abgespeichert.