Aufgabe 1

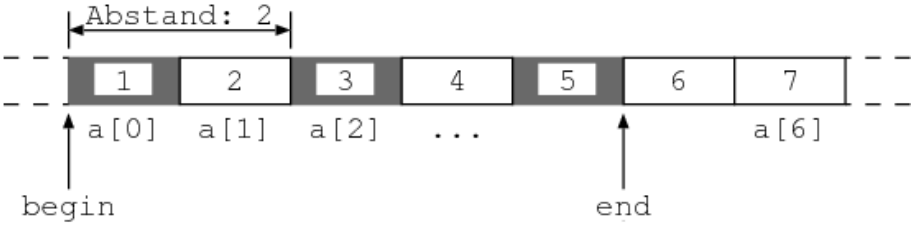
Ein eindimensionales Teilfeld (subarray) kann unter Bezugnahme auf ein bereits existierendes Feld `a` durch folgende Angaben beschrieben werden:

- Zeiger auf das erste Element des Teilfeldes
- Zeiger auf das Ende des Teilfeldes
- Abstand zwischen zwei aufeinander folgenden Elementen

Die folgende Skizze zeigt ein auf einem Feld

```
int a[] = {1,2,3,4,5,6,7};
```

definiertes Teilfeld, das durch die Zeiger `begin` und `end`, sowie den Abstand 2 festgelegt ist und die Werte 1, 3 und 5 beinhaltet:



1. Vereinbaren Sie einen Datentyp `subarray_t` zum Speichern von Teilfeldern, die auf Feldern mit Elementen des Typs **`char`** (oder alternativ einem beliebigen Elementtyp `element_t`) basieren. Ein Objekt dieses Typs soll keine Kopie des Basisfeldes beinhalten.

2. Schreiben Sie eine Funktion, die ein Teilfeld in einem Objekt des Typs `subarray_t` abspeichert, indem es die drei Werte (z.B. `begin`, `end`, `2`), die das Teilfeld beschreiben, in das Objekt einträgt.

3. Schreiben Sie eine Funktion, die alle Elemente eines übergebenen Teilfeldes auf der Standardausgabe ausgibt.

4. Implementieren Sie eine Funktion

   ```
   char *getpointer(subarray_t sa, int i);
   ```

   `getpointer` soll einen Zeiger auf das `i`-te Element des Teilfeldes `sa` zurückgeben.

5. Implementieren Sie eine Funktion

   ```
   char getelement(subarray_t sa, int i);
   ```

   `getelement` soll den Inhalt des `i`-ten Elements des Teilfeldes `sa` liefern.

6. Erstellen Sie eine Header-Datei, die alle in den Teilaufgaben 1. - 5. realisierten Datentypen/ Funktionen einem Anwendungsprogramm verfügbar machen soll.

7. Schreiben Sie ein Hauptprogramm, das eine 3×3-Matrix mit den Werten

```
1  2  3
4  5  6
7  8  9
```

initialisiert. Zunächst soll ein Teilfeld erzeugt werden, das die Diagonale dieser Matrix repräsentiert. Die Diagonalelemente sind dann unter Verwendung der Funktionen `getpointer` und `getelement` zu negieren.

Anschließend soll in einer Schleife jede Spalte der Matrix als Teilfeld des Typs `subarray_t` abgebildet und mit Hilfe der in 3. erstellten Funktion ausgegeben werden.

Aufgabe 2 *(Zusatzaufgabe, Bearbeitung freiwillig)*
Um eine Nachricht mit Hilfe der sogenannten *hebräischen Methode* zu verschlüsseln, wird sie in eine Matrix zeilenweise eingetragen und spaltenweise wieder ausgelesen. Verwendet man z.B. eine 5×4-Matrix, um den Text

`Kernighan Ritchie`

zu verschlüsseln, dann ergibt sich die folgende Matrix:

K	e	r	n
i	g	h	a
n		R	i
t	c	h	i
e			

Durch spaltenweises Auslesen der Matrix erhält man die verschlüsselte Nachricht:

`Kinteeg c rhRh naii`

Schreiben Sie eine Funktion, die den Text einer Zeichenkette auf diese Weise verschlüsselt, **ohne** jedoch **die Zeichenkette** tatsächlich in eine Matrix **umzuspeichern**. Ihre Funktion soll also ohne den zusätzlichen Speicherplatz für die Matrix auskommen, indem die einzelnen Spalten der Matrix nacheinander als Teilfelder (subarrays) über der gegebenen Zeichenkette definiert werden und diese in den verschlüsselten Resultatstring kopiert werden.
Die zu schreibende Funktion soll mit den folgenden Argumenten aufgerufen werden:

- die Zeichenkette mit der zu verschlüsselnden Nachricht.
- die Zeilen- und Spaltenzahl der zu verwendenden Verschlüsselungs-Matrix.
- die Adresse eines Zeichenvektors, in den die verschlüsselte Nachricht einzutragen ist.

Als Resultatwert soll Ihre Funktion einen Zeiger auf die verschlüsselte Nachricht zurückgeben.