Aufgabe 1: Personendatenbank

Entwickeln Sie eine Personendatenbank. Zu jeder Person gespeichert werden soll:
- Anrede (enum)
- Vorname (char*)
- Nachname (char*)
- Strasse (char*)
- Postleitzahl (int)
- Ort (char*)

Die einzelnen Datensätze sollen in einem Array vom passenden Typ und von fester Größe gespeichert werden. Entwickeln Sie Funktionen, die das
- Anlegen bzw. Hinzufügen einer Person
- Löschen einer Person
- Auflisten aller belegten Einträge des Arrays auf dem Bildschirm

ermöglichen. Weiterhin soll Ihr Programm mit einer textuellen Menusteuerung versehen sein.
Beim Anlegen einer Person sollen die Eigenschaften Anrede, Vorname, etc. vom Benutzer abgefragt werden. Lesen Sie die Benutzereingaben zunächst in einen Puffer fester Länge ein. Erzeugen Sie daraus die Zeichenkette in der passenden Länge (Sie wissen vorher nicht, wie lang z.B. ein Name ist, den der Benutzer eingibt. Andererseits wollen Sie auch keinen Speicher verschwenden.)
Beim Auflisten sollen alle belegten Einträge angezeigt werden. Ein Eintrag gilt als belegt, wenn die Postleitzahl ungleich 0 ist. Gelöschte Einträge haben somit die Postleitzahl 0. Beim Löschen eines Eintrages in dem Array sollen die nachfolgenden Einträge also nicht vorrücken. Die Ausgabe soll zusätzlich den Array-Index eines jeden Eintrags enthalten. Die Löschfunktion soll den Index des zu löschenden Eintrags als Eingabe erhalten.
Achten Sie bei der Implementierung auf die korrekte Benutzung von Zeichenketten (Speicher, Länge, abschließendes Nullbyte, etc.).
Testen Sie das Programm (was passiert bei unerwarteter Eingabe, was passiert beim Löschen nichtexistierender Einträge, etc.).
Hinweis: `structs, Arrays, printf(3), scanf(3), strlen(3), strdup(3), strcpy(3), malloc(3), free(3)`

Aufgabe 2: Verkette Listen

Schreiben Sie ein Programm, das eine verkettete Liste von Zahlen implementiert.

Hinweis: Erstellen Sie ein `struct Element`, welches eine Zahl und einen Pointer auf das nächste Element enthält.

Desweiteren sollen folgende Funktionen implementiert werden:

```
void add(struct Element *e, int i);          //Fügt ein Element hinzu
int remove(struct Element *e, int position);      //Löscht das Element an der
Stelle "position" und gibt es zurück
int get(struct Element *e, int position);    //Gibt die Zahl, die im Element an der
Stelle "position" gespeichert ist, zurück
```

Empfohlene Aufgabe:

Aufgabe 3: playfair

Die Playfair-Verschlüsselung ist ein 1854 von Charles Wheatstone (ja, der mit der Messbrücke) erfundenes Verschlüsselungsverfahren, bei dem jedes Buchstabenpaar des Klartextes durch ein anderes Buchstabenpaar ersetzt wird. Bekannt wurde das Verfahren dann unter dem Namen eines Bekannten von Wheatstone, Lyon Playfair.

Der Algorithmus funktioniert wie folgt:

1. Erstellen einer 5*5-Matrix

Tragen Sie das Schlüsselwort zeilenweise in eine 5*5-Matrix ein. Tragen Sie jeden Buchstaben nur einmal in die Matrix ein. Füllen Sie anschließend die freien Plätze mit den noch fehlenden Buchstaben des lateinischen Alphabets der Reihe nach aus. Verzichten Sie auch auf den Buchstaben 'j'. Jeder Buchstabe des Alphabets, ausser 'j', kommt also genau einmal in der Matrix vor.

2. Zerlegen des Klartextes

Zerlegen Sie den Klartext in Gruppen zu je zwei Buchstaben. Entfernen Sie dabei Leer- und Satzzeichen, ersetzen Sie 'j' durch 'i'. Kommen in einer Gruppe zwei gleiche Buchstaben vor, z.B. 'mm', fügen Sie dieser Gruppe ein Füllzeichen ('x') hinzu, und zerlegen Sie erneut. 'mm' wird also zu 'mx' und die nächste Gruppe fängt mit dem überzähligen 'm' an. Der Rest des Klartextes wird wie gewohnt weiter zerlegt. Bleibt für die letzte Gruppe nur ein Einzelbuchstabe über, wird diese Gruppe auch mit 'x' zu einer Zweiergruppe aufgefüllt.

3. Codierung

Zur Codierung (und Decodierung) wird die Matrix aus Schritt 1 herangezogen.
Bei der Codierung sind drei Fälle zu beachten:

- Beide Buchstaben der betrachteten Buchstabengruppe liegen in der selben Reihe in der Matrix:
  Jeder Buchstabe wird verschlüsselt, indem er durch den nächstfolgenden der selben Zeile ersetzt wird. Handelt es sich beim Klartextbuchstaben um den letzten der Zeile, wird mit dem Ersten der Zeile verschlüsselt.
- Beide Buchstaben der Buchstabengruppe liegen in der selben Spalte:
  Jeder Buchstabe wird verschlüsselt, indem er durch den unter ihm stehenden der selben Spalte ersetzt wird. Handelt es sich beim Klartextbuchstaben um den Untersten der Spalte, wird er mit dem Obersten der Spalte verschlüsselt.
- Beide Buchstaben der Buchstabengruppe liegen weder in der selben Reihe noch in der selben Spalte:
  Man geht in der Zeile des ersten Klarbuchstaben nach rechts oder links zur Spalte des zweiten Buchstaben. Der dort stehende Buchstabe ist die Chiffre für diesen. Mit dem zweiten Buchstaben wird ebenso verfahren.

4. Decodierung

Die Entschlüsselung erfolgt ebenso, nur mit umgekehrter Bewegungsrichtung in der Matrix. Statt des nachfolgenden Buchstabens wird der vorangehende ausgewählt, statt des darunter stehenden wird der darüber genommen. Allein im dritten Fall kann genauso vorgegangen werden wie bei der Codierung.

5. Beispiel

Das Schlüsselwort sei `'extrawurst'`. Damit ergibt sich die folgende Matrix:

```
E X T R A
W U S B C
D F G H I
K L M N O
P Q V Z Y
```

Wenn wir damit den Text `'c macht spass'` verschlüsseln, ergibt sich der Text `'so ci gr wv tc ut'`.

Implementieren Sie die Codierung und Decodierung. Überlegen Sie sich Testfälle und implementieren Sie diese.