1. Schreiben Sie eine Funktion `char to_upper_case(char c)`, die einen kleinen Buchstaben in den entsprechenden großen Buchstaben umwandelt (ohne Verwendung der Standardbibliothek).

2. Schreiben Sie eine Funktion `void upper(char text[])`, die alle Buchstaben in einer Zeichenkette in Großbuchstaben umwandelt (ohne Umlaute). Alle Sonderzeichen und Ziffern sollen nicht verändert werden. Verwenden Sie an geeigneter Stelle die Funktion `char to_upper_case(char c)`. Rufen Sie die Funktion `upper` auf und geben Sie das Ergebnis aus. Hinweis: In C gibt es eine enge Verbindung zwischen Zeigern und Feldern. Eine Feldvariable steht für die Speicheradresse, ab der das Feld (Array) beginnt. Bei einem Funktionsaufruf wird damit ein Feld *by reference* übergeben.

3. Implementieren Sie eine *Queue* (Warteschlange) mit einem Array. Die *Queue* soll die beiden Funktionen `enqueue` und `dequeue` besitzen. Mit `void enqueue(int i)` wird die Zahl i in die *Queue* aufgenommen. Wenn die Queue voll ist, dann wird der Wert am Ende der *Queue* einfach überschrieben. Mit `int dequeue()` wird die Zahl, die am längsten in der *Queue* ist, logisch aus der *Queue* genommen und zurückgeliefert. Wenn die *Queue* leer ist, dann wird -1 zurückgeliefert. Damit der Speicherplatz im Array möglichst gut genutzt wird, realisieren Sie einen logischen Ringspeicher (Sie dürfen die Modulorechnung verwenden). Wenn das Ende des Arrays erreicht ist, dann wird vorne im Array weitergearbeitet, wenn sich dort nicht der Anfang der Warteschlange befindet. Ansonsten wird der Wert am Ende der *Queue* überschrieben. Testen Sie die *Queue* mit dem folgenden Programm. Arbeiten Sie dabei mit einer Arraylänge von 2 für die Queue. Die Implementierung soll aber natürlich für beliebige aber feste Arraylängen funktionieren.

```
#include <stdio.h>
#include "queue.h"

main(){
  enqueue(1);
  enqueue(2);
  enqueue(2);
  printf("%i ", dequeue());
  enqueue(3);
  printf("%i ", dequeue());
  printf("%i ", dequeue());
  printf("%i ", dequeue());
  enqueue(4);
  enqueue(5);
  printf("%i ", dequeue());
  printf("%i \n", dequeue());
}
```

Die Ausgabe lautet dann:

```
1 2 3 -1 4 5
```