Aufgabe 1 (20P Punkte) zum 02.05.2013

Implementieren Sie eine Freispeicherverwaltung mit der Schnittstelle

```
/* allocate.h */
void* allocate();
void deallocate(void *data);
```

die den Speicher aus einem festen Speicherblock entsprechend der folgenden Deklarationen erhält:

```
#define BLOCKSIZE 40
#define NUM_BLOCKS 1024

extern unsigned char arena[BLOCKSIZE*NUM_BLOCKS];
extern unsigned short allocated_map[NUM_BLOCKS/16];
```

Implementieren Sie Ihre Lösung in der Bibliothek libarena.
Die Funktion allocate() alloziert Blöcke fester Größe (BLOCKSIZE); die Funktion deallocate() gibt diese Blöcke wieder frei.

Falls alle Blöcke alloziert sind, gibt allocate() als Ergebnis 0. Das Feld allocated_map speichert mit einem Bit pro Block, welche Blöcke alloziert sind.

Senden Sie Ihre Lösung in Form eines einzelen gzip-komprimierten Tarfiles ein. Dieses sollte im Wurzelverzeichnis ein Makefile haben, mit zwei Zielen "liballocate.a" und "testapp" (basierend auf testapp.c). Gehen Sie in Ihrer Lösung davon aus, dass die Makefile-Variable LIBARENA das Verzeichnis angibt, in dem sich libarena.a und arena.h befinden.

Zusatzaufgabe: Erweitern Sie Ihre Speicherverwaltung auf mehrere Arena-Blöcke. Verwenden Sie malloc()/free() zum Anlegen der Arenas. Implementieren Sie dazu die Funktionen

`void * newArena( int blocksize, int numblocks)` und
`freeArena(void * arena).`

Die Funktionen

`allocateEx( void * arena )` und
`deallocateEx( void * arena, void * data)`

sollen wie allocate() und deallocate() funktionieren, und zusätzlich einen Verweis auf die zu verwendene Arena enthalten.
Zur Verwaltung der verschiedenen Arenas wird eine einfach-verkettete Liste empfohlen.