```
/*                                                                  tab:8
 *
 * mem220.h - header file for ECE220's simple memory management package
 *
 * "Copyright (c) 2003-2018 by Steven S. Lumetta."
 *
 * Permission to use, copy, modify, and distribute this software and its
 * documentation for any purpose, without fee, and without written agreement is
 * hereby granted, provided that the above copyright notice and the following
 * two paragraphs appear in all copies of this software.
 *
 * IN NO EVENT SHALL THE AUTHOR OR THE UNIVERSITY OF ILLINOIS BE LIABLE TO
 * ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
 * DAMAGES ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION,
 * EVEN IF THE AUTHOR AND/OR THE UNIVERSITY OF ILLINOIS HAS BEEN ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * THE AUTHOR AND THE UNIVERSITY OF ILLINOIS SPECIFICALLY DISCLAIM ANY
 * WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE SOFTWARE
 * PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND NEITHER THE AUTHOR NOR
 * THE UNIVERSITY OF ILLINOIS HAS ANY OBLIGATION TO PROVIDE MAINTENANCE,
 * SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS."
 *
 * Author:          Steve Lumetta
 * Version:         2
 * Creation Date:   4 December 2003
 * Filename:        mem220.h
 * History:
 *      SL      1       4 December 2003
 *              First written.
 *      SL      2       28 March 2018
 *              Updated slightly for new numbering and standards.
 */

#if !defined(_MEM220_H)
#define _MEM220_H

#include <stdint.h>


/*
   These constants define the limitations on memory allocation with
   the package.  Nothing larger can be compiled.  Note that the code for
   the package must be recompiled if these numbers are changed.
*/

#define MEM220_MAX_ALLOC_LOG  20
#define MEM220_MAX_ALLOC      (1UL << MEM220_MAX_ALLOC_LOG)

/*
   mem220_allocate

   Allocates n_bytes and returns a pointer to the new memory.  If no memory
   is available, or if 0 bytes are requested, returns NULL.  Note that the
   new memory may contain arbitrary values.
*/
void* mem220_allocate (size_t n_bytes);

/*
   mem220_allocate_and_zero

   Allocates n_bytes, fills the new memory with zeroes, and returns a
   pointer to the new memory.  If no memory is available, or if 0 bytes
   are requested, returns NULL.
*/
void* mem220_allocate_and_zero (size_t n_bytes);

/*
   mem220_reallocate

   Attempts to change the size of a previously allocated block of memory.
   The parameters passed are a pointer to the pointer to the old block
   (possibly NULL, if no previous block existed) and the new desired size.
   If possible, a new block of the appropriate size is allocated, any
   data in the old block are copied into the new block, the old block
   is freed, the pointer is changed, and 0 is returned.  If the allocation
   of a new block fails, the pointer to the old block is not changed,
   the old block (if it existed) is not freed, and -1 is returned.
*/
int32_t mem220_reallocate (void** ptr_to_ptr, size_t n_bytes);

/*
    mem220_free

    Returns control of a block of memory to the memory management system.
    The block should not be accessed after a call to mem220_free.  The
    block may be returned by a successive call to any of the allocation
    functions.
*/
void mem220_free (void* ptr);

#endif /* !defined(_MEM220_H) */
```