







Implement a simple memory allocator based on the so-called "Buddy System" scheme, that allocates memory in blocks with sizes that are powers of two, starting at a block size that is given as an argument when the allocator is initialized.

\* The memory allocator shall be implemented as a C module my\_allocator, which consists of a header file my\_allocator.h and my\_allocator.c. It should provide the functions my\_malloc() and my\_free(), very similar to the UNIX calls malloc() and free().

\* Evaluate the correctness (up to some point) and the performance of your allocator. For this use the given strange implementation of a highly-recursive Ackermann function. In this implementation of the Ackermann function, random blocks of memory are allocated and de-allocated sometime later, generating a large combination of different allocation patterns.

\* Write a program called memtest, which reads the basic block size and the memory size (in bytes) from the command line, initializes the memory, and then calls the Ackermann function. It measures the time it takes to perform the number of memory operations. Make sure that the program exits cleanly if aborted (using atexit() to install the exit handler).

\* Use the getopt() C library function to parse the command line for arguments. The synopsis of the memtest program is of the form

 memtest [-b <blocksize>] [-s <memsize>]

 -b <blocksize> defines the block size, in bytes. Default is 128 bytes.

 -s <memsize> defines the size of the memory to be allocated, in bytes.

Default is 512kB.

\* Repeatedly invoke the Ackermann function with increasingly larger values for n and m (be careful to keep n <= 3; the processing time increases very steeply for larger values of n).

\* Make sure that the allocator gets de-allocated (and its memory freed) when the program either exits or aborts (for example, when the user presses Ctrl-C).