Chris Fretz
Professor Parashar
Systems Programming
12/5/14

Malloc/Free Implementation

- System Overview
  - Memory Layout
    - Block Layout
      - Each block in memory consists of header, data, and footer sections.
      - Each header consists of first either a char, short, int, or size_t (the type depends on the size of the heap. The default of 5000 can be represented with a short) containing the size of the data segment, followed by a hash value (used for detecting valid blocks. Hash value is based off of the machine's host name) stored in a short. A short was used for the hash value to minimize memory footprint, and, although not foolproof, it provides more than a 99.99% likelihood of successfully identifying invalid memory segments (only a 1 in 65,536 chance of coming across the correct bit pattern accidentally. Giving each block a unique hash would have required too much memory overhead). No incorrect behavior was detected during testing.
      - Each data segment varies in size based on what the user requested. Size of data segment is stored in the header and footer.
      - Footer consists of either a char, short, int, or size_t (depending on size of heap) containing the size of the block.
      - Implementation enforces that all allocated segments are sized along multiples of two, so the lowest order bit of the size field in both the header and footer is used as a flag to indicate whether or not the block is in use.
    - Implementation uses a static, global, char array as the heap as instructed. Heap is broken up into blocks of varying size based on need. Immediately following initialization, heap consists of a blank footer (indicating a size of zero) padding the beginning and end of the array (used for detecting the end of the array in the merge and backtrack routines. In truth, I could have just used the array pointer and its size, but initially I was trying to avoid declaring it globally. It didn't work out, and I didn't bother changing things since it still works) and a single, unused, block of memory taking up all remaining space. Block is subdivided based on need.
  - Memory Management
    - Allocation
      - Upon receiving a request for an allocation, the find_best_fit routine steps through the heap using the headers as links, and returns a pointer to the block closest to the requested size. In case of a tie, routine chooses the first option it found.
      - After finding the best fit, allocate checks how much, if any, extra memory the block contains, and splits said block using the routine split_block if the extraneous memory accounts for more than 1% of the total heap size. Allocate then advances the block pointer to the data section and returns it.
      - If no suitable block is found, allocate returns a NULL pointer.
    - Deallocation
      - Upon receiving a request for deallocation, deallocate first checks to make sure the memory segment has been initialized, then checks to make sure the pointer is in range. Assuming these checks pass, it decrements the given pointer back to the header segment of the block, and then checks both the hash value and usage status of the block. Assuming all aforementioned checks pass, deallocate proceeds to mark the block as unused and begins consolidation of the heap, otherwise it prints an error and terminates execution.

Chris Fretz
Professor Parashar
Systems Programming
12/5/14

- Consolidation
  - The consolidation process consists of two distinct steps (encapsulated in two different functions): backtracking and merging.
  - Backtracking
    - Using the footer of the previous block as a link, the backtrack routine traverses backwards through the heap until it finds a block in use or reaches the beginning of the heap, at which point it returns a pointer to the most recent, unused, block.
  - Merging
    - Starting with the block returned by backtrack, the merge function begins merging forward, rewriting headers and footers as it goes, until it encounters either a block in use or the end of the heap. At the end of merge, all consecutively unused blocks behind the given block will be merged into one, unused, block.