malloclab-reference **mm-reference.c**

```
 */
void *mm_malloc(size_t size)
{
    size_t asize;      // Allocated block size
    block_t *block;
    void *bp = NULL;

    if (size == 0) // Ignore spurious request
        return bp;

    // Adjust block size to include overhead and to meet alignment requirements
    asize = round_up(size + dsize, dsize);

    // Search the free list for a fit
    block = find_fit(asize);

    // If no fit is found, return NULL
    if (block == NULL)
        return bp;

    // Mark block as allocated
    size_t block_size = get_size(block);
    write_header(block, block_size, true);
    write_footer(block, block_size, true);

    // Try to split the block if too large
    split_block(block, asize);

    bp = header_to_payload(block);

    return bp;
}


/* Free allocated block */
void mm_free(void *bp)
{
    if (bp == NULL)
        return;

    block_t *block = payload_to_header(bp);
    size_t size = get_size(block);

    // The block should be marked as allocated
    if (!get_alloc(block)) {
        fprintf(stderr, "ERROR.  Attempted to free unallocated block\n");
        exit(1);
    }

    // Mark the block as free
    write_header(block, size, false);
    write_footer(block, size, false);

    // Try to coalesce the block with its neighbors
    coalesce_block(block);
}

/* Print status of every block in heap */
void mm_status(FILE *fp) {
    block_t *block = heap_start;
    while (block != heap_end) {
        fprintf(fp, "  Block address %p,  size = %zd, allocated = %s\n",
                block, get_size(block), get_alloc(block) ? "Y" : "N");
        block = find_next(block);
    }

}

/******** The remaining content below are helper and debug routines ********/


/*
 * Attempt to coalesce block with its predecessor and successor
 */
static void coalesce_block(block_t *block)
{

    size_t size = get_size(block);

    block_t *block_next = find_next(block);
    block_t *block_prev = find_prev(block);

    bool prev_alloc = extract_alloc(*find_prev_footer(block));
    bool next_alloc = get_alloc(block_next);

    if (prev_alloc && next_alloc)              // Case 1
    {
        // Nothing to do
    }
```