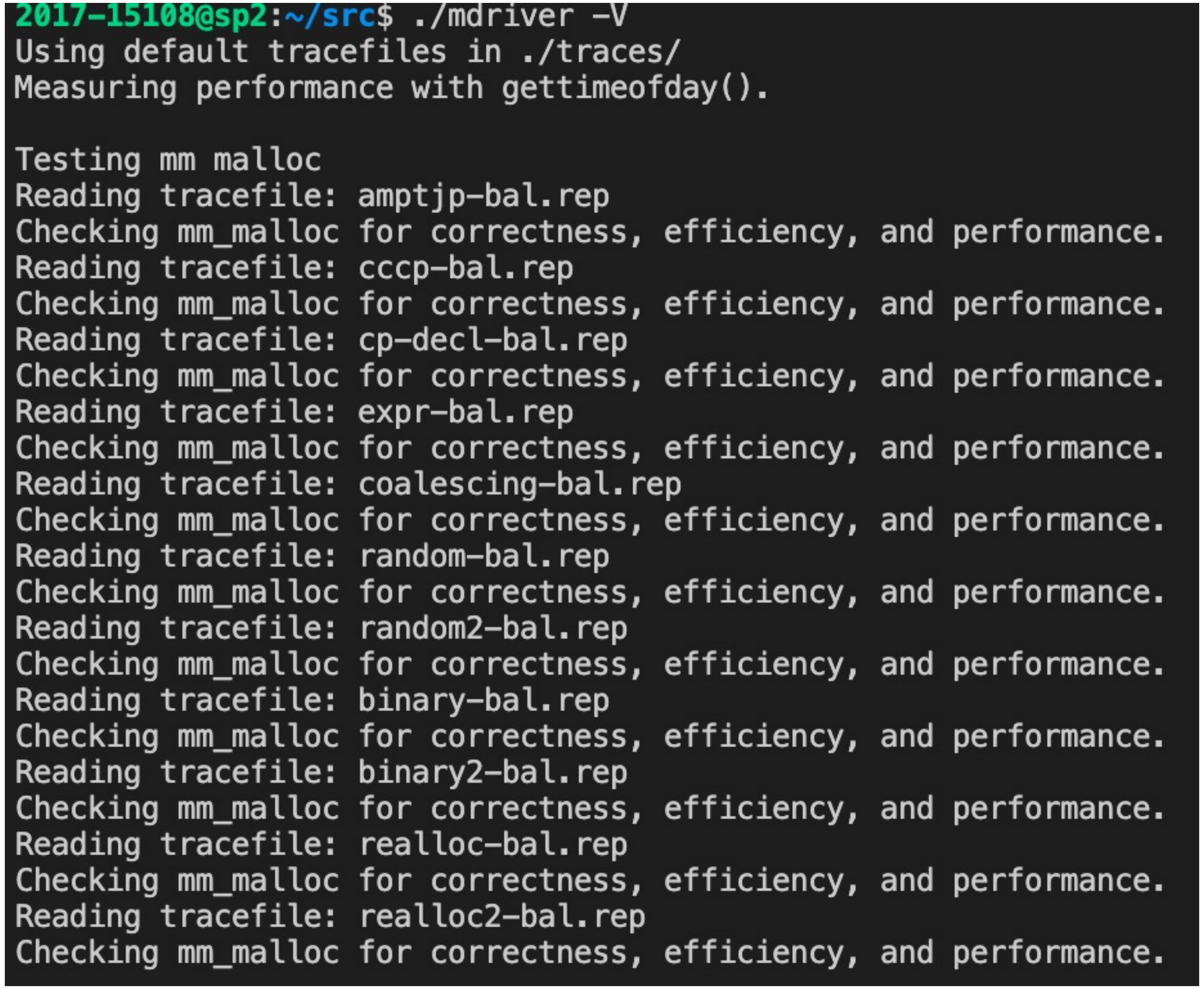
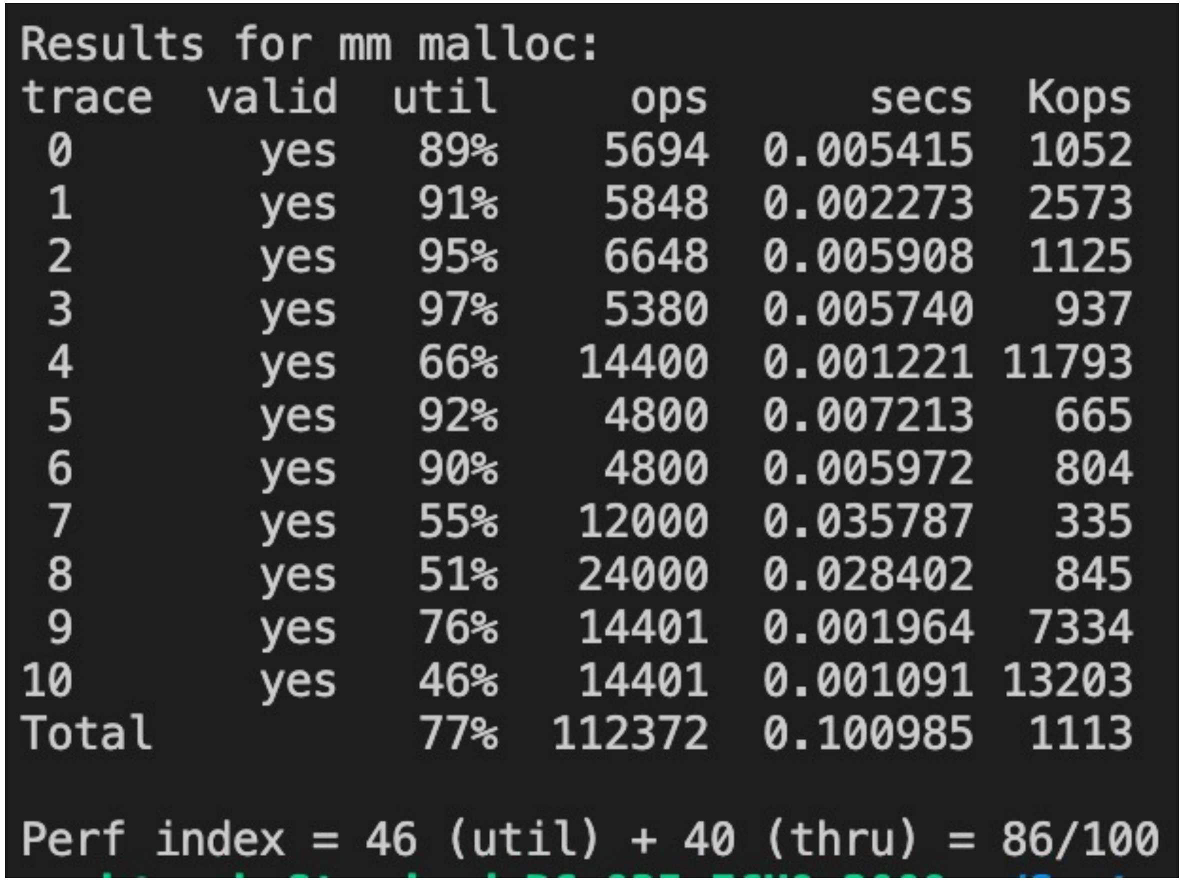
시스템 프로그래밍 Lab3 Report

경영학과 2017-15108
박지상

1. 실행결과

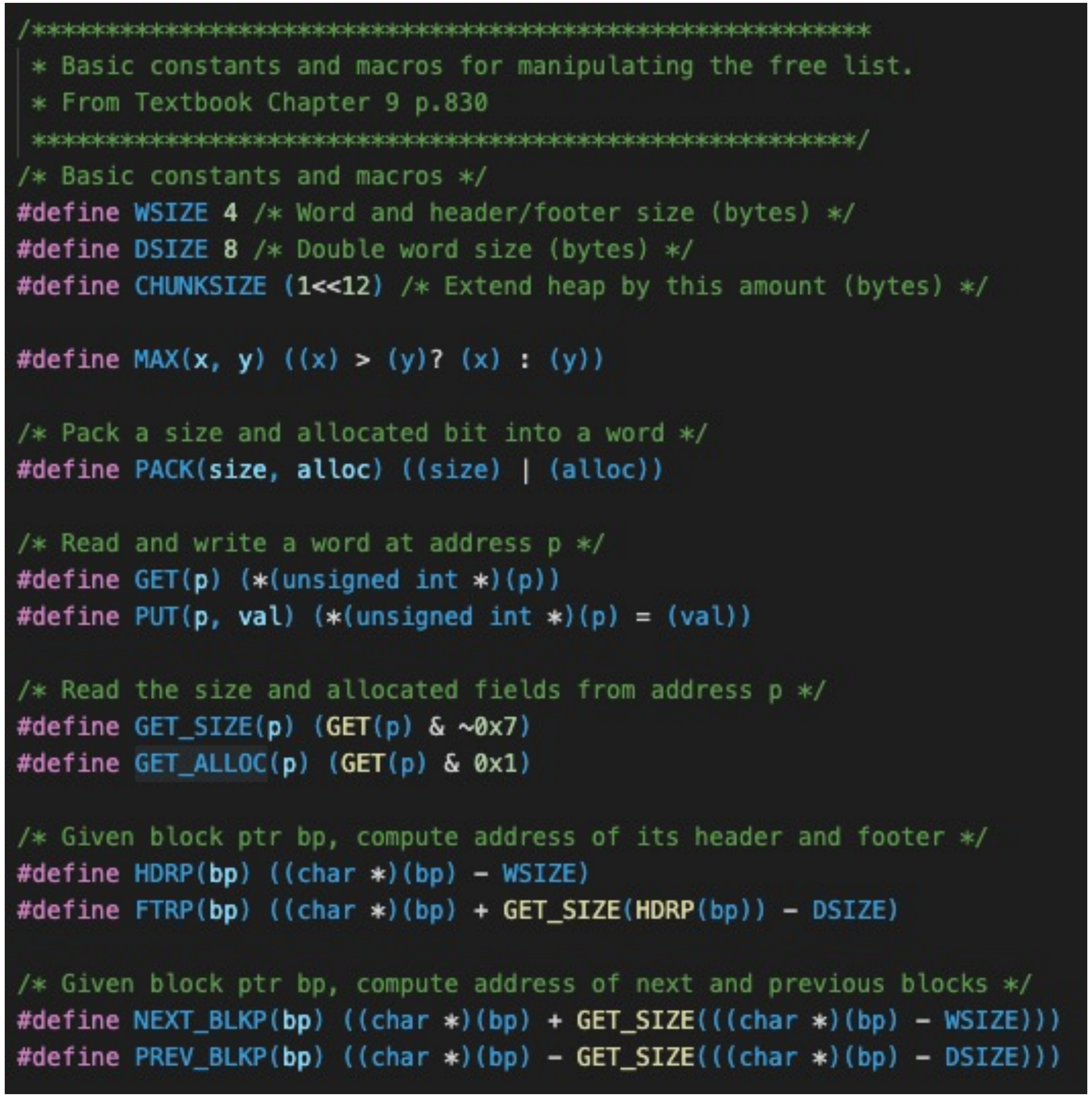


```
2017-15108@sp2:~/src$ ./mdriver -V
Using default tracefiles in ./traces/
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: amptjp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cccp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cp-decl-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: expr-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: coalescing-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
```

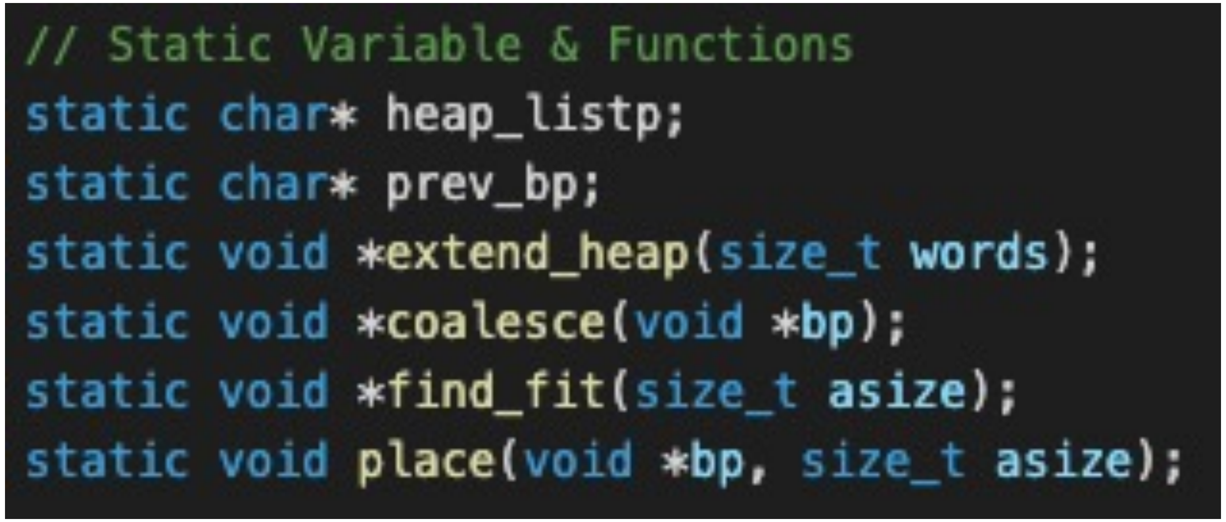


```
Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   89%    5694  0.005415  1052
 1       yes   91%    5848  0.002273  2573
 2       yes   95%    6648  0.005908  1125
 3       yes   97%    5380  0.005740   937
 4       yes   66%   14400  0.001221 11793
 5       yes   92%    4800  0.007213   665
 6       yes   90%    4800  0.005972   804
 7       yes   55%   12000  0.035787   335
 8       yes   51%   24000  0.028402   845
 9       yes   76%   14401  0.001964  7334
10       yes   46%   14401  0.001091 13203
Total          77%  112372  0.100985  1113

Perf index = 46 (util) + 40 (thru) = 86/100
```

2. 구현 방법 : Constants & Macros



```
/************************************************************
 * Basic constants and macros for manipulating the free list.
 * From Textbook Chapter 9 p.830
 ************************************************************/
/* Basic constants and macros */
#define WSIZE 4 /* Word and header/footer size (bytes) */
#define DSIZE 8 /* Double word size (bytes) */
#define CHUNKSIZE (1<<12) /* Extend heap by this amount (bytes) */

#define MAX(x, y) ((x) > (y)? (x) : (y))

/* Pack a size and allocated bit into a word */
#define PACK(size, alloc) ((size) | (alloc))

/* Read and write a word at address p */
#define GET(p) (*(unsigned int *)(p))
#define PUT(p, val) (*(unsigned int *)(p) = (val))

/* Read the size and allocated fields from address p */
#define GET_SIZE(p) (GET(p) & ~0x7)
#define GET_ALLOC(p) (GET(p) & 0x1)

/* Given block ptr bp, compute address of its header and footer */
#define HDRP(bp) ((char *)(bp) - WSIZE)
#define FTRP(bp) ((char *)(bp) + GET_SIZE(HDRP(bp)) - DSIZE)

/* Given block ptr bp, compute address of next and previous blocks */
#define NEXT_BLKP(bp) ((char *)(bp) + GET_SIZE(((char *)(bp) - WSIZE)))
#define PREV_BLKP(bp) ((char *)(bp) - GET_SIZE(((char *)(bp) - DSIZE)))
```



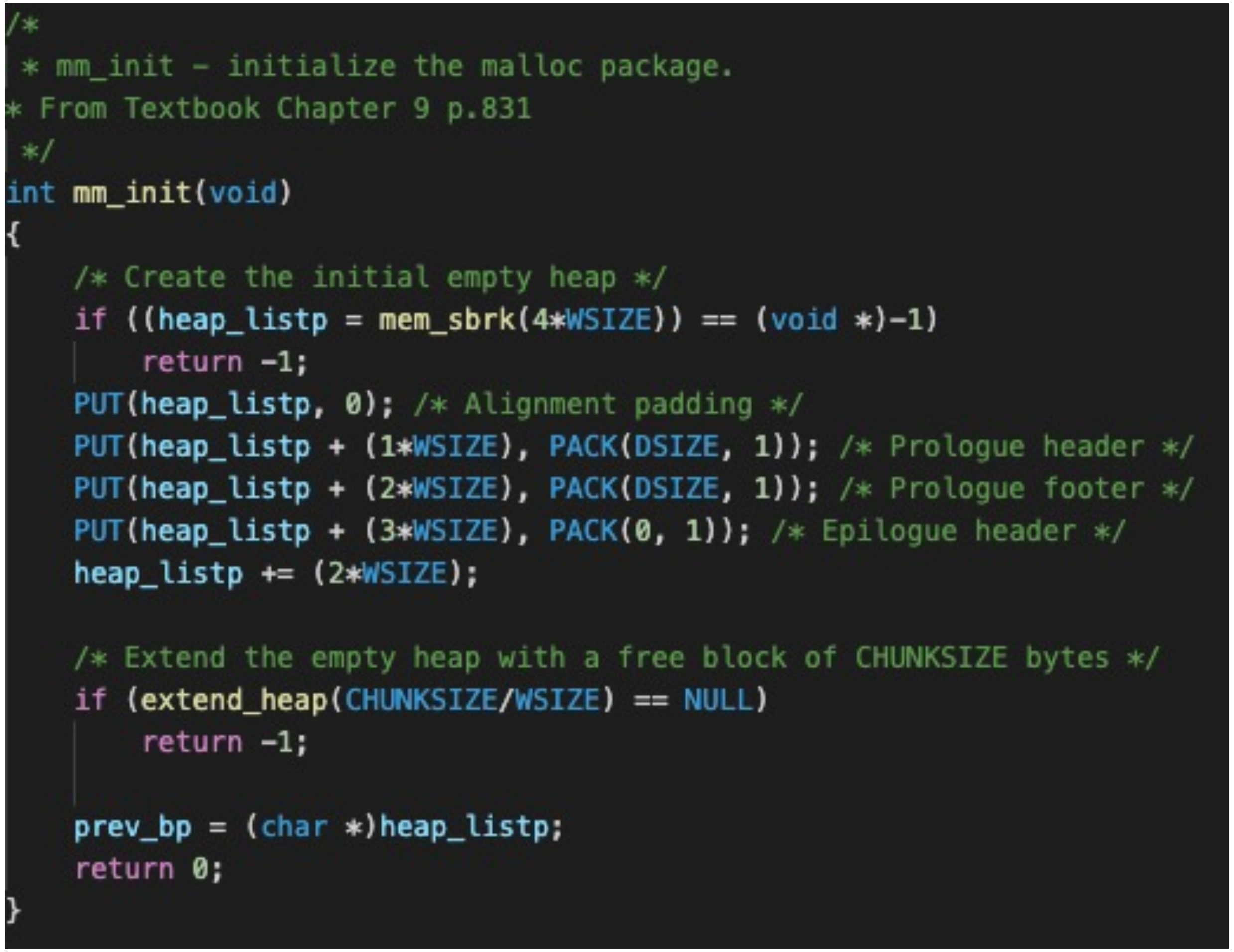
```
// Static Variable & Functions
static char* heap_listp;
static char* prev_bp;
static void *extend_heap(size_t words);
static void *coalesce(void *bp);
static void *find_fit(size_t asize);
static void place(void *bp, size_t asize);
```

교과서에 제시된 Constants와 Macros를 추가하였다.

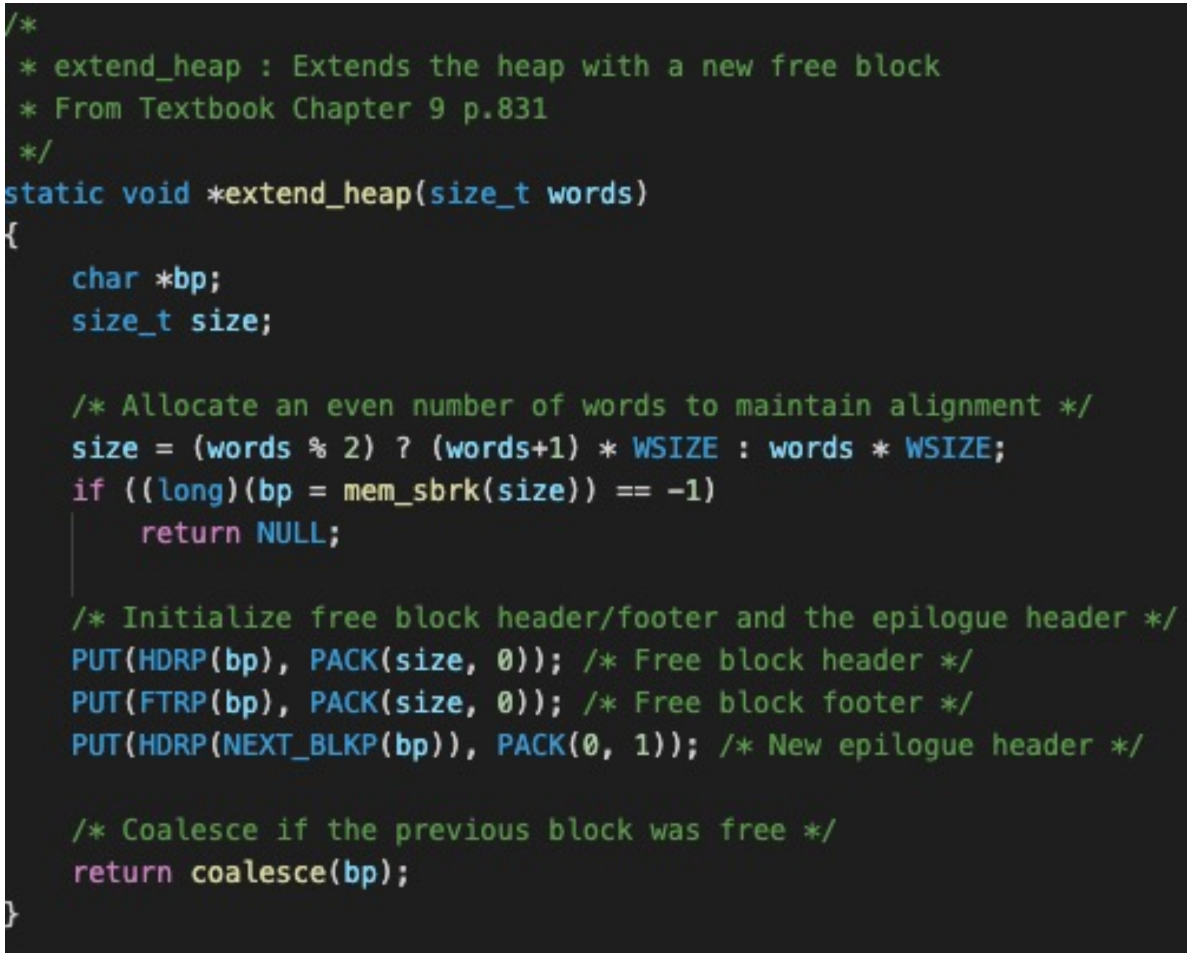
Global variable로 다음을 선언하였다.

- char* heap_listp: heap block의 root node
- char* prev_bp: next-fit을 구현하기 위한 기록

2. 구현 방법 : mm_init(), extend_heap()



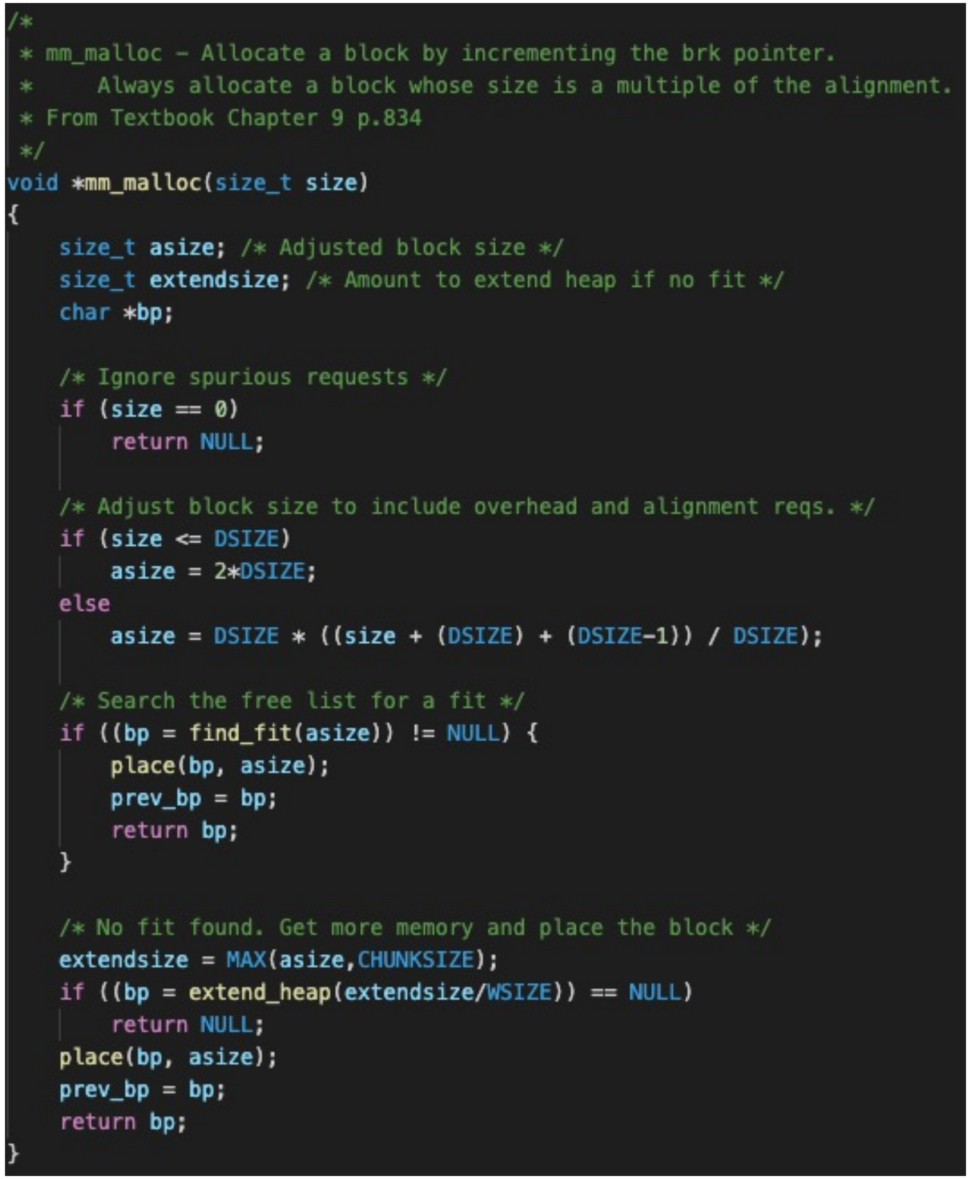
```c
/*
 * mm_init - initialize the malloc package.
* From Textbook Chapter 9 p.831
 */
int mm_init(void)
{
    /* Create the initial empty heap */
    if ((heap_listp = mem_sbrk(4*WSIZE)) == (void *)-1)
        return -1;
    PUT(heap_listp, 0); /* Alignment padding */
    PUT(heap_listp + (1*WSIZE), PACK(DSIZE, 1)); /* Prologue header */
    PUT(heap_listp + (2*WSIZE), PACK(DSIZE, 1)); /* Prologue footer */
    PUT(heap_listp + (3*WSIZE), PACK(0, 1)); /* Epilogue header */
    heap_listp += (2*WSIZE);

    /* Extend the empty heap with a free block of CHUNKSIZE bytes */
    if (extend_heap(CHUNKSIZE/WSIZE) == NULL)
        return -1;

    prev_bp = (char *)heap_listp;
    return 0;
}
```



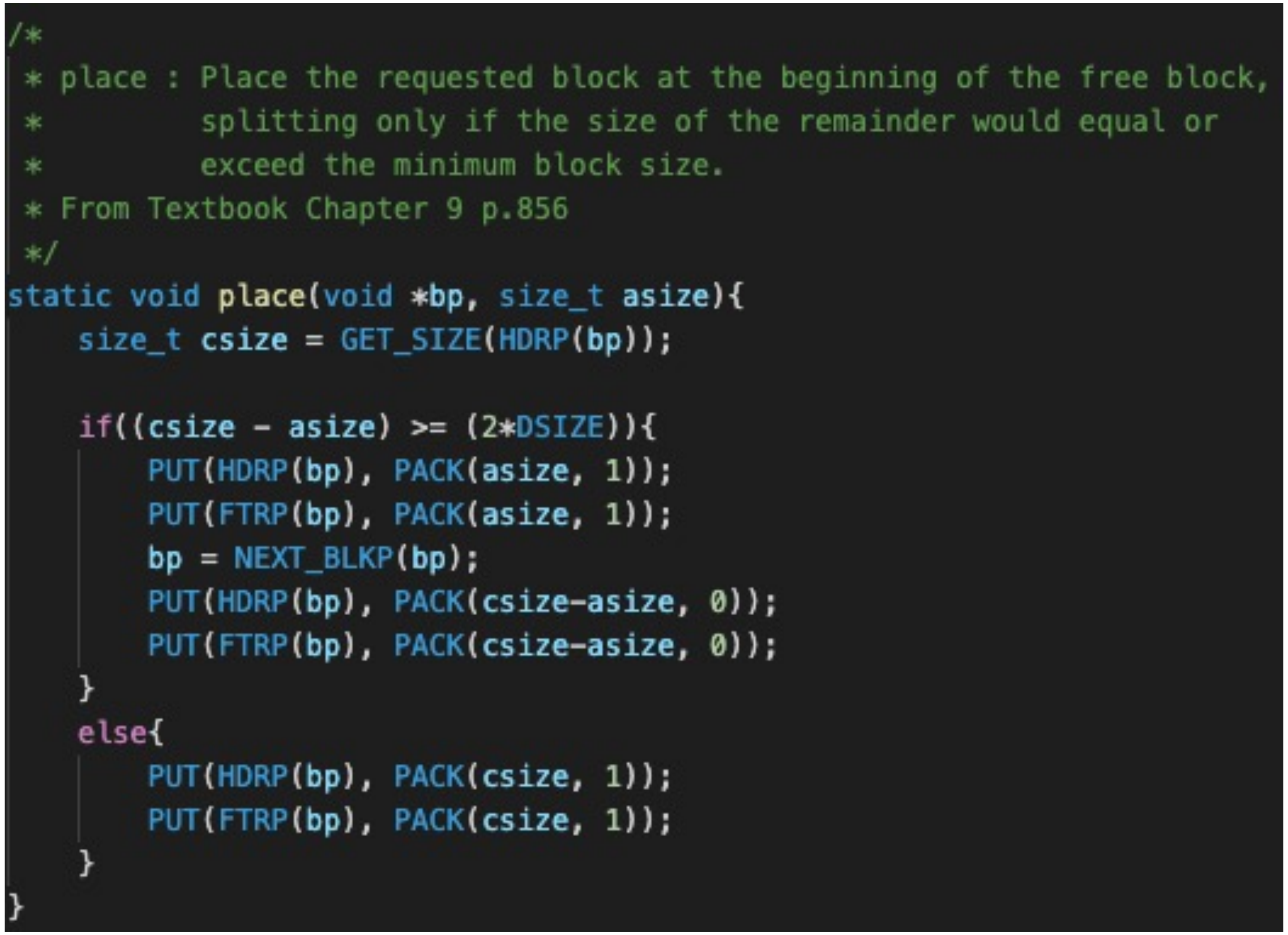
```c
/*
 * extend_heap : Extends the heap with a new free block
 * From Textbook Chapter 9 p.831
 */
static void *extend_heap(size_t words)
{
    char *bp;
    size_t size;

    /* Allocate an even number of words to maintain alignment */
    size = (words % 2) ? (words+1) * WSIZE : words * WSIZE;
    if ((long)(bp = mem_sbrk(size)) == -1)
        return NULL;

    /* Initialize free block header/footer and the epilogue header */
    PUT(HDRP(bp), PACK(size, 0)); /* Free block header */
    PUT(FTRP(bp), PACK(size, 0)); /* Free block footer */
    PUT(HDRP(NEXT_BLKP(bp)), PACK(0, 1)); /* New epilogue header */

    /* Coalesce if the previous block was free */
    return coalesce(bp);
}
```

Textbook에 나와있는 방식으로 구현함

2. 구현 방법 : mm_malloc(), place()



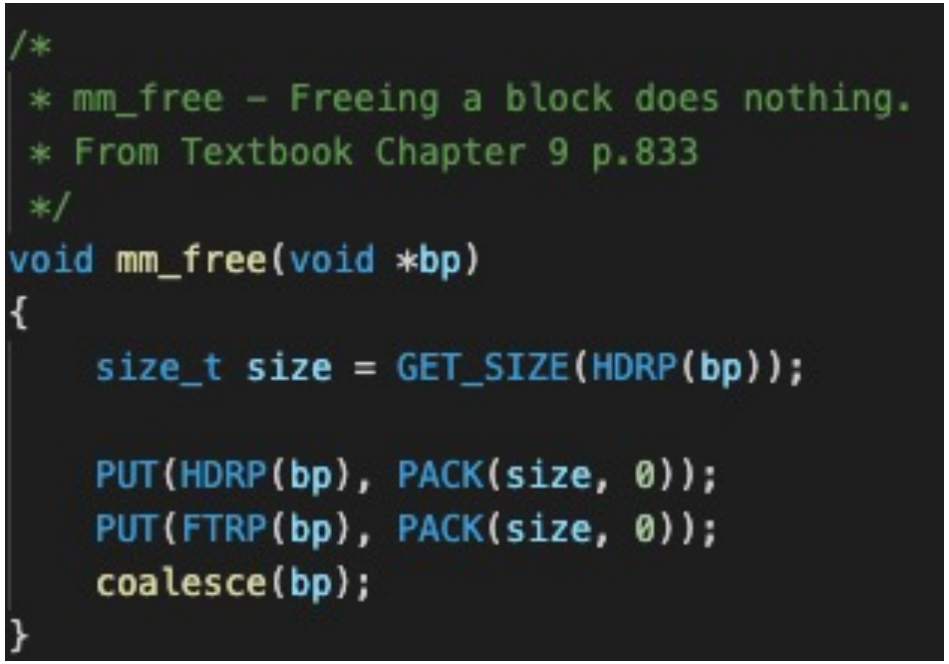
```
/*
 * mm_malloc - Allocate a block by incrementing the brk pointer.
 *     Always allocate a block whose size is a multiple of the alignment.
 * From Textbook Chapter 9 p.834
 */
void *mm_malloc(size_t size)
{
    size_t asize; /* Adjusted block size */
    size_t extendsize; /* Amount to extend heap if no fit */
    char *bp;

    /* Ignore spurious requests */
    if (size == 0)
        return NULL;

    /* Adjust block size to include overhead and alignment reqs. */
    if (size <= DSIZE)
        asize = 2*DSIZE;
    else
        asize = DSIZE * ((size + (DSIZE) + (DSIZE-1)) / DSIZE);

    /* Search the free list for a fit */
    if ((bp = find_fit(asize)) != NULL) {
        place(bp, asize);
        prev_bp = bp;
        return bp;
    }

    /* No fit found. Get more memory and place the block */
    extendsize = MAX(asize,CHUNKSIZE);
    if ((bp = extend_heap(extendsize/WSIZE)) == NULL)
        return NULL;
    place(bp, asize);
    prev_bp = bp;
    return bp;
}
```



```
/*
 * place : Place the requested block at the beginning of the free block,
 *         splitting only if the size of the remainder would equal or
 *         exceed the minimum block size.
 * From Textbook Chapter 9 p.856
 */
static void place(void *bp, size_t asize){
    size_t csize = GET_SIZE(HDRP(bp));

    if((csize - asize) >= (2*DSIZE)){
        PUT(HDRP(bp), PACK(asize, 1));
        PUT(FTRP(bp), PACK(asize, 1));
        bp = NEXT_BLKP(bp);
        PUT(HDRP(bp), PACK(csize-asize, 0));
        PUT(FTRP(bp), PACK(csize-asize, 0));
    }
    else{
        PUT(HDRP(bp), PACK(csize, 1));
        PUT(FTRP(bp), PACK(csize, 1));
    }
}
```

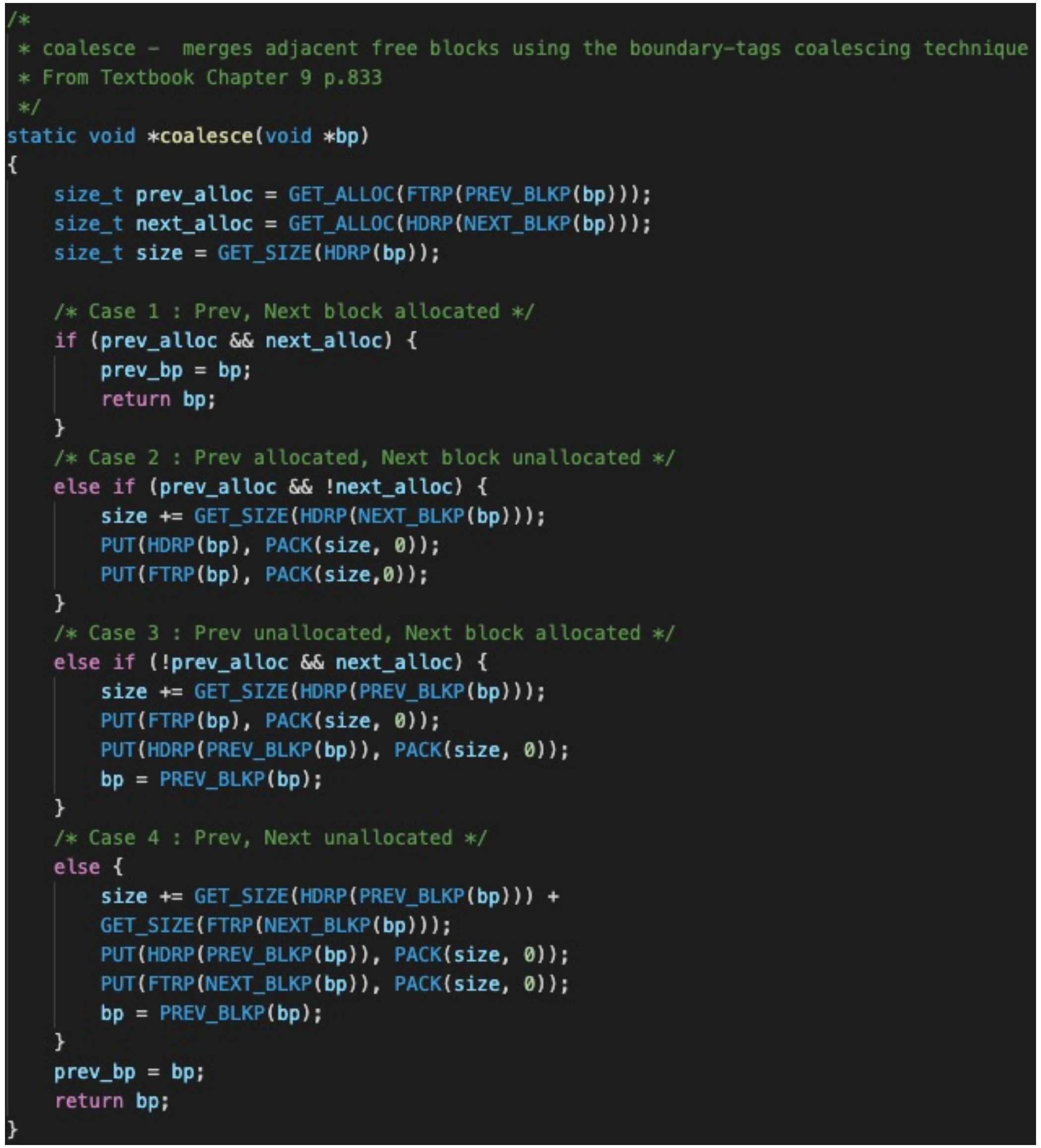
Textbook에 나와있는 방식으로 구현함

2. 구현 방법 : mm_free(), coalesce()



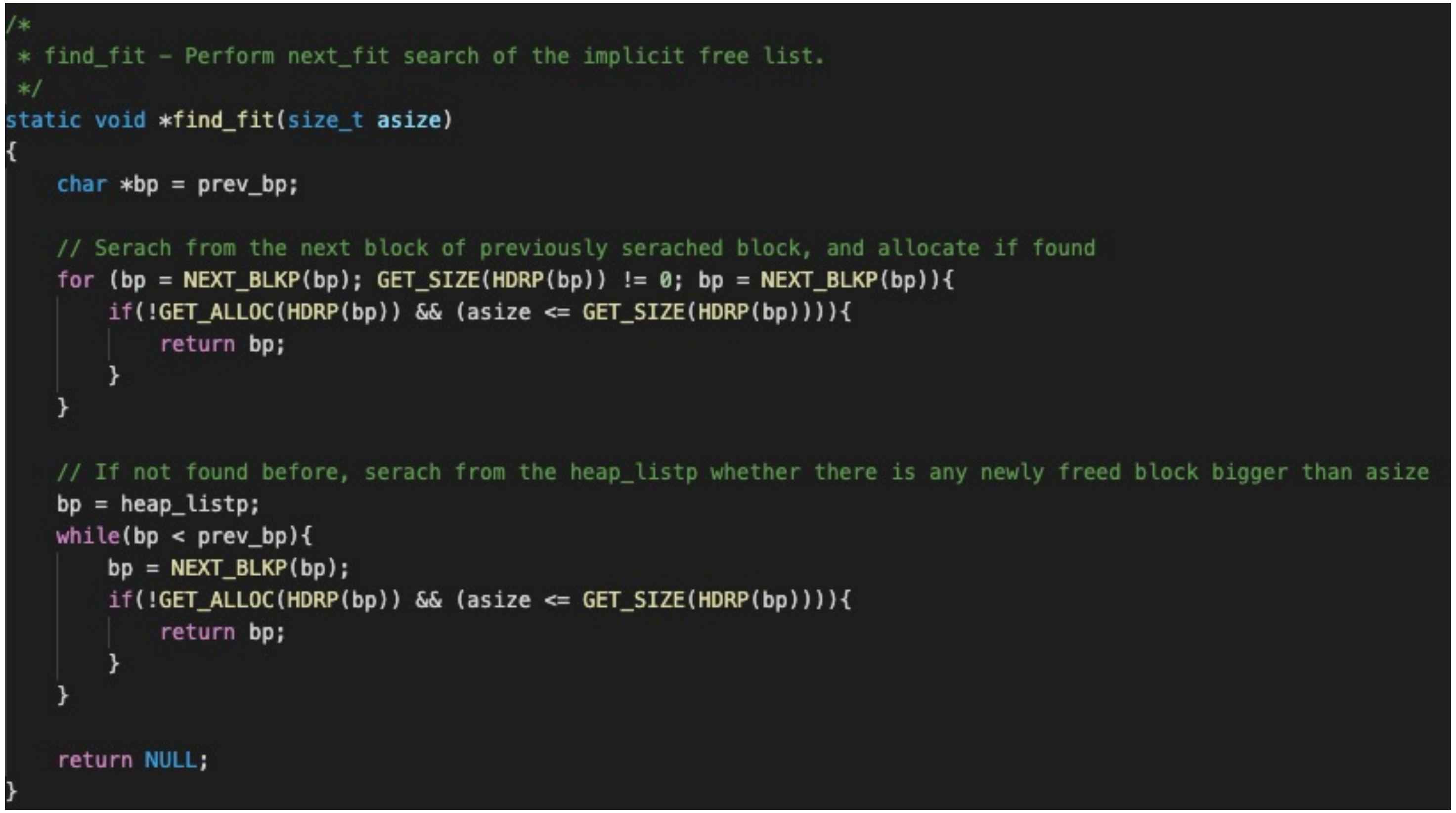
```
/*
 * mm_free - Freeing a block does nothing.
 * From Textbook Chapter 9 p.833
 */
void mm_free(void *bp)
{
    size_t size = GET_SIZE(HDRP(bp));

    PUT(HDRP(bp), PACK(size, 0));
    PUT(FTRP(bp), PACK(size, 0));
    coalesce(bp);
}
```

Textbook에 나와있는 방식으로 구현함



```
/*
 * coalesce -  merges adjacent free blocks using the boundary-tags coalescing technique
 * From Textbook Chapter 9 p.833
 */
static void *coalesce(void *bp)
{
    size_t prev_alloc = GET_ALLOC(FTRP(PREV_BLKP(bp)));
    size_t next_alloc = GET_ALLOC(HDRP(NEXT_BLKP(bp)));
    size_t size = GET_SIZE(HDRP(bp));

    /* Case 1 : Prev, Next block allocated */
    if (prev_alloc && next_alloc) {
        prev_bp = bp;
        return bp;
    }
    /* Case 2 : Prev allocated, Next block unallocated */
    else if (prev_alloc && !next_alloc) {
        size += GET_SIZE(HDRP(NEXT_BLKP(bp)));
        PUT(HDRP(bp), PACK(size, 0));
        PUT(FTRP(bp), PACK(size,0));
    }
    /* Case 3 : Prev unallocated, Next block allocated */
    else if (!prev_alloc && next_alloc) {
        size += GET_SIZE(HDRP(PREV_BLKP(bp)));
        PUT(FTRP(bp), PACK(size, 0));
        PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0));
        bp = PREV_BLKP(bp);
    }
    /* Case 4 : Prev, Next unallocated */
    else {
        size += GET_SIZE(HDRP(PREV_BLKP(bp))) +
        GET_SIZE(FTRP(NEXT_BLKP(bp)));
        PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0));
        PUT(FTRP(NEXT_BLKP(bp)), PACK(size, 0));
        bp = PREV_BLKP(bp);
    }
    prev_bp = bp;
    return bp;
}
```

2. 구현 방법 : find_fit()



```
/*
 * find_fit - Perform next_fit search of the implicit free list.
 */
static void *find_fit(size_t asize)
{
    char *bp = prev_bp;

    // Serach from the next block of previously serached block, and allocate if found
    for (bp = NEXT_BLKP(bp); GET_SIZE(HDRP(bp)) != 0; bp = NEXT_BLKP(bp)){
        if(!GET_ALLOC(HDRP(bp)) && (asize <= GET_SIZE(HDRP(bp)))){
            return bp;
        }
    }

    // If not found before, serach from the heap_listp whether there is any newly freed block bigger than asize
    bp = heap_listp;
    while(bp < prev_bp){
        bp = NEXT_BLKP(bp);
        if(!GET_ALLOC(HDRP(bp)) && (asize <= GET_SIZE(HDRP(bp)))){
            return bp;
        }
    }

    return NULL;
}
```

Textbook에 나와있는 first_fit 방식으로는 성능이 낮게 나와 next-fit으로 구현함.

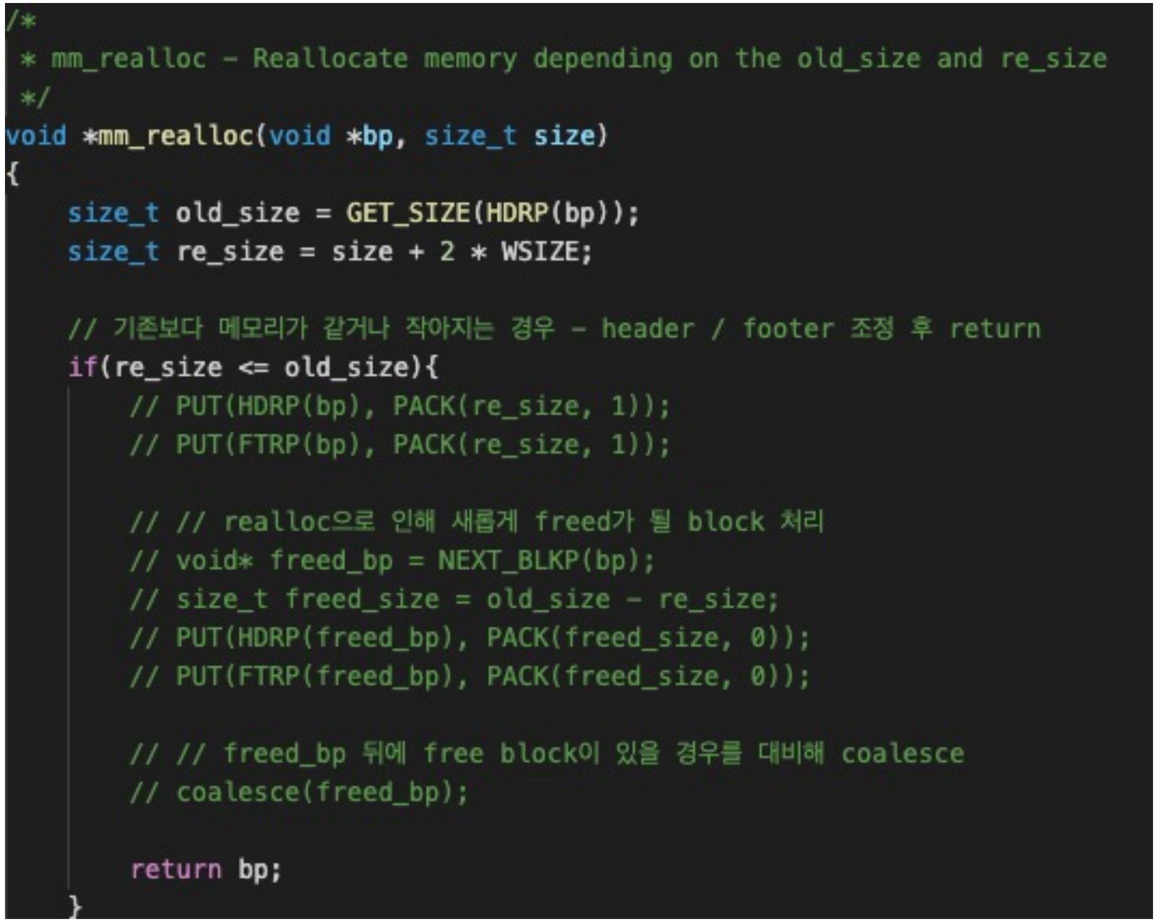
next-fit을 구현하기 위해 static variable로 char* prev_bp를 선언하고
다음과 같은 부분에서 init과 update했다.

1) mm_init()
prev_bp = (char *)heap_listp;에서 Init

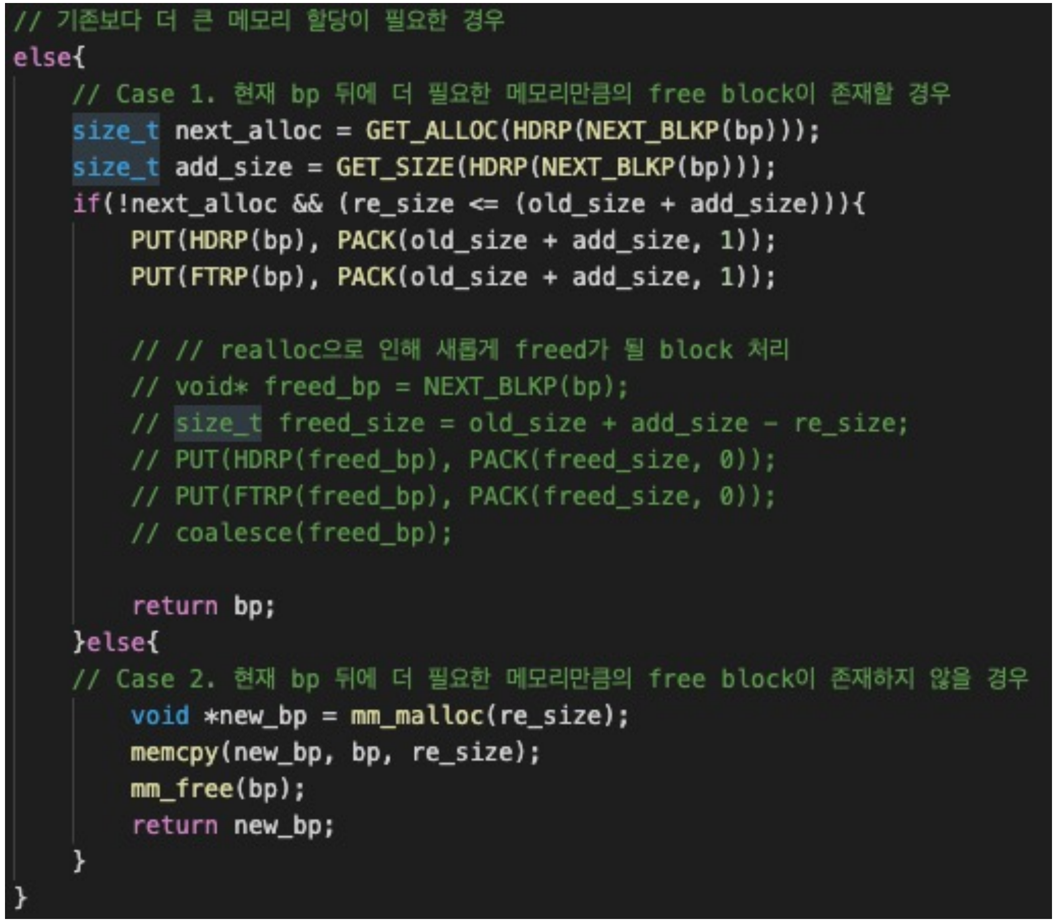
2) mm_malloc()
place(bp, asize)후 prev_bp = bp;

3) coalesce()
coalesce가 끝난 후 prev_bp=bp;

2. 구현 방법 : mm_realloc()



```c
/*
 * mm_realloc - Reallocate memory depending on the old_size and re_size
 */
void *mm_realloc(void *bp, size_t size)
{
    size_t old_size = GET_SIZE(HDRP(bp));
    size_t re_size = size + 2 * WSIZE;

    // 기존보다 메모리가 같거나 작아지는 경우 - header / footer 조정 후 return
    if(re_size <= old_size){
        // PUT(HDRP(bp), PACK(re_size, 1));
        // PUT(FTRP(bp), PACK(re_size, 1));

        // // realloc으로 인해 새롭게 freed가 될 block 처리
        // void* freed_bp = NEXT_BLKP(bp);
        // size_t freed_size = old_size - re_size;
        // PUT(HDRP(freed_bp), PACK(freed_size, 0));
        // PUT(FTRP(freed_bp), PACK(freed_size, 0));

        // // freed_bp 뒤에 free block이 있을 경우를 대비해 coalesce
        // coalesce(freed_bp);

        return bp;
    }
```



```c
// 기존보다 더 큰 메모리 할당이 필요한 경우
else{
    // Case 1. 현재 bp 뒤에 더 필요한 메모리만큼의 free block이 존재할 경우
    size_t next_alloc = GET_ALLOC(HDRP(NEXT_BLKP(bp)));
    size_t add_size = GET_SIZE(HDRP(NEXT_BLKP(bp)));
    if(!next_alloc && (re_size <= (old_size + add_size))){
        PUT(HDRP(bp), PACK(old_size + add_size, 1));
        PUT(FTRP(bp), PACK(old_size + add_size, 1));

        // // realloc으로 인해 새롭게 freed가 될 block 처리
        // void* freed_bp = NEXT_BLKP(bp);
        // size_t freed_size = old_size + add_size - re_size;
        // PUT(HDRP(freed_bp), PACK(freed_size, 0));
        // PUT(FTRP(freed_bp), PACK(freed_size, 0));
        // coalesce(freed_bp);

        return bp;
    }else{
    // Case 2. 현재 bp 뒤에 더 필요한 메모리만큼의 free block이 존재하지 않을 경우
        void *new_bp = mm_malloc(re_size);
        memcpy(new_bp, bp, re_size);
        mm_free(bp);
        return new_bp;
    }
}
```

기존의 block size와 새롭게 allocate하려는 size를 비교해서 각 case에 따른 처리를 해주었다.

다만, realloc 과정에서 발생하는 freed block에 대한 처리를 하려하였으나, seg fault가 지속적으로 발생하여 결국 해결방법을 찾지 못하였다.

3. 어려웠던 점

- dynamically allocated block의 구조와 header / footer의 구조에 대한 이해가 필요했다
- Dynamically allocated block을 처리하는 데 필요한 macro에 대한 명확한 이해가 필요했다.
- Next-fit을 구현하는 데 있어서 prev_bp를 init하고 update하는 지점에 대해서 고민해야했다.
- Realloc()을 구현할 때 중간에 발생하는 freed-block에 대한 처리를 시도하였으나,

  지속적으로 알 수 없는 seg fault가 발생했다.

4. 새롭게 배운 점

- Macro를 통해서 복잡한 작업을 간편하게 표현할 수 있음을 알게되었다.

- mm_init()을 할 때, alignment padding과 prologue header/footer, epilogue header가 필요함을 알게되었다.

- malloc을 할 때 block size를 adjust해야함을 배웠다.

- Lab01에서처럼 단순하게 free()후 malloc()하는 방식의 realloc()이 아니라,

  기존에 할당되었던 size와 새롭게 할당하려는 size에 따라 case를 나누어 더 효율적으로 구현할 수 있다.