System Programming Project 3

담당 교수 : 김영재 교수님

이름 : 안도현

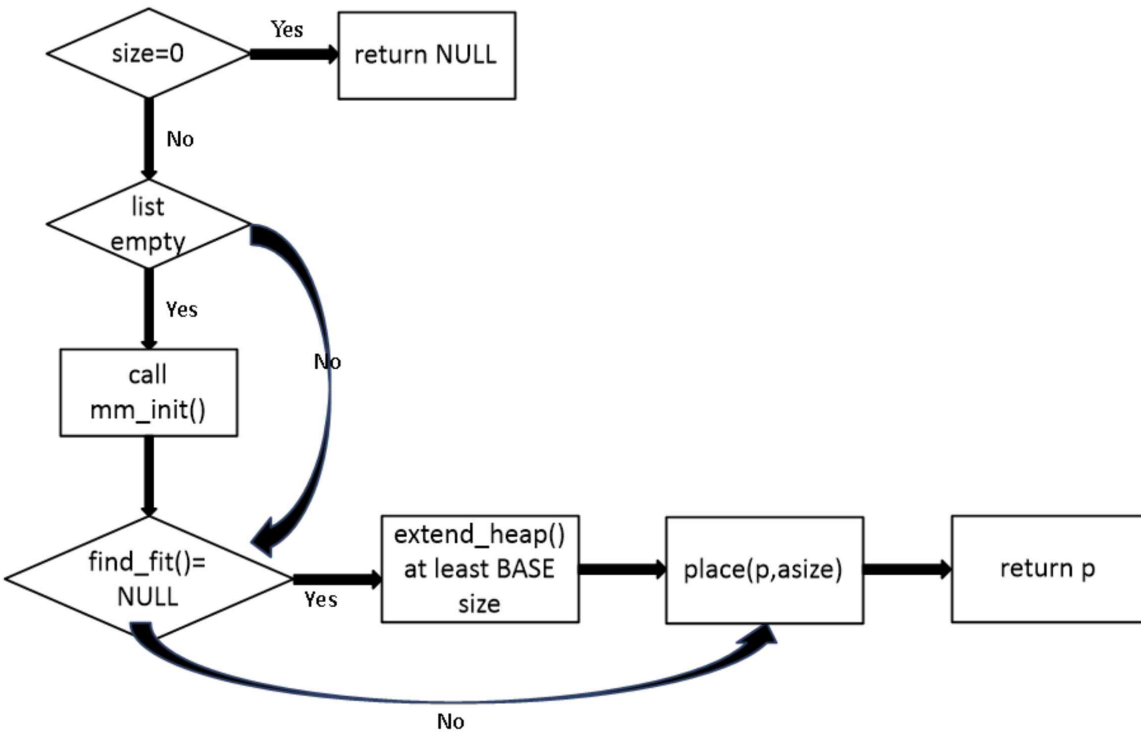
학번 : 20181650

1. **Design(all function included)**

-mm_init

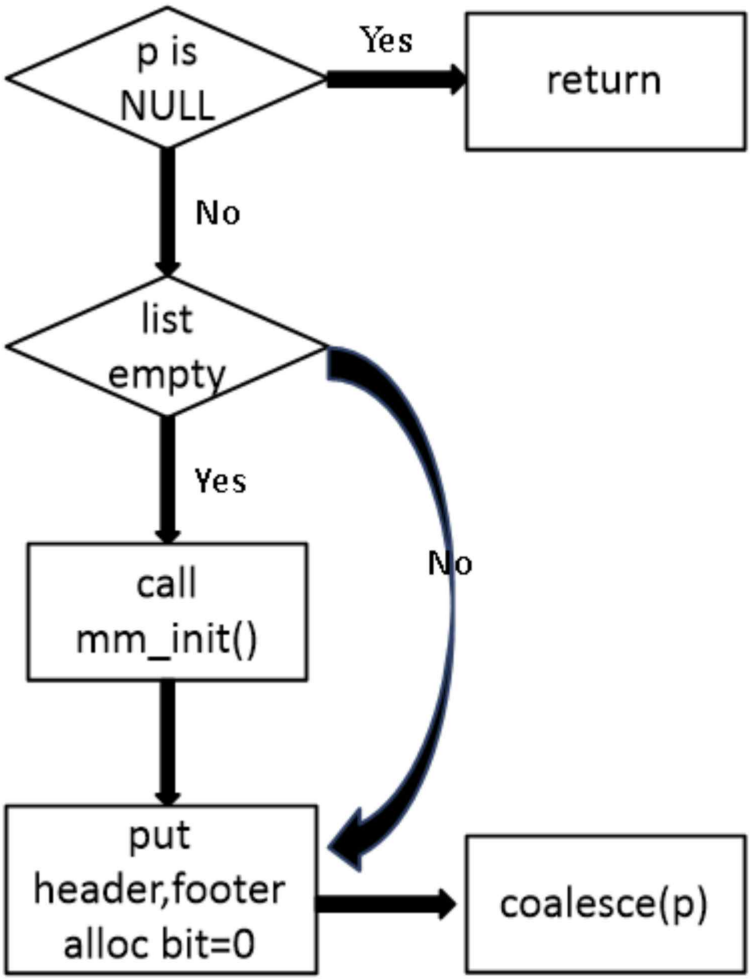
imp_list_start(list의 시작점)에 16byte를 할당하여 base header, footer, 및 epilogue footer를 넣어준다. 이 epilogue footer는 리스트의 끝을 알리기 위해 꼭 필요한 부분이다. nowpos는 Next fit search에서 search의 시작점이 될 변수로 처음에는 리스트의 처음 지점으로 준다. 그리고 처음에 BASE 만큼의 공간을 할당해놓고 프로그램을 시작한다.

-mm_malloc



size가 0이면 NULL을 리턴하고, mm_init()이 진행되지 않은 상태라면 호출하여 힙의 기본 구성을 할당한다. 이후 find_fit() 함수를 통해 맞는 free block이 존재하는지 확인하고, 있으면 그대로 place 함수를 통해 할당, 없다면 extend_heap 함수를 통해 추가적인 공간을 확보하여 할당한다. 그리고 그 베이스 포인터 p를 리턴한다.

-mm_free()



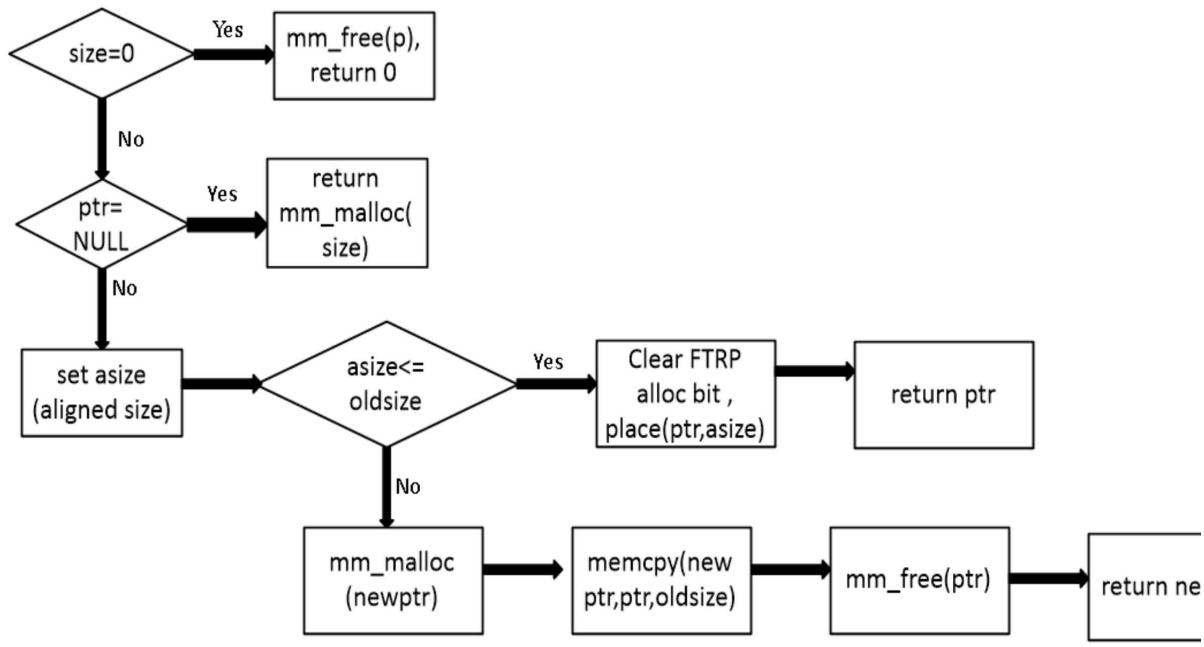
역시 null 포인터를 free 하려고 하면 그냥 바로 리턴하고, mm_init이 호출된 적이 없다면 호출하고, free하려는 블록의 header와 footer의 tag에 최하위 비트를 0으로 설정한 뒤 이웃한 free block과 합치기 위해 coalesce 함수를 호출한다.

-coalesce()

CASE 1	PREV_ALLOC && NEXT_ALLOC. Right away return bp
CASE 2	PREV_ALLOC && !NEXT_ALLOC. coalesce with previous block.
CASE 3	!PREV_ALLOC && NEXT_ALLOC. coalesce with next block.
CASE 4	!PREV_ALLOC && !NEXT_ALLOC. coalesce with previous and next block.

coalesce는 기본적으로 이전과 다음 블락의 free 여부를 얻어 그들과 합치는 함수이다. 합쳤을 때 기준으로 가장 왼쪽 블락의 헤더에 총 사이즈와 alloc bit를 pack해 넣어주며, 같은 것을 footer에도 다시 넣어주어 그 base pointer을 리턴하게 되는 함수이다. 추가적으로 next fit search를 하기 때문에 nowpos가 블록의 중간을 가리킨 채로 남아서 문제가 생기지 않도록 위치를 조정도 한다.

-mm_realloc()



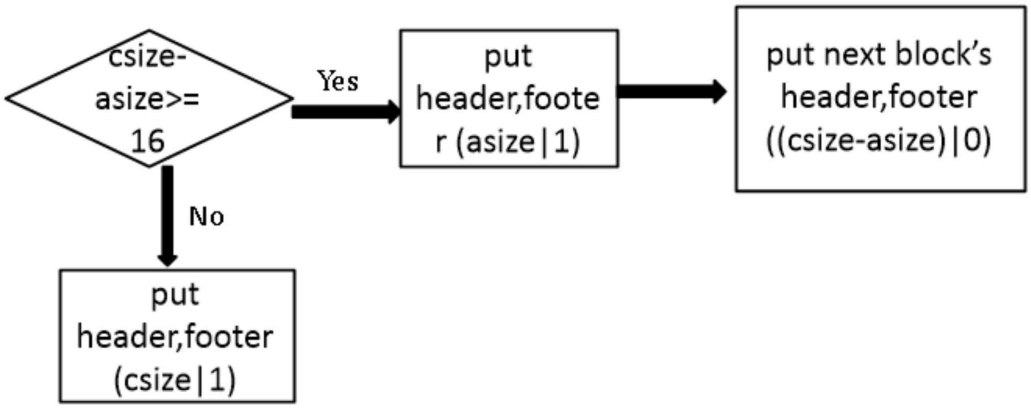
realloc은 우선 size와 ptr을 체크하여 size가 0이면 free, ptr이 null이면 malloc를 호출하는 것으로 함수를 대체한다. 그것이 아니라면 alignment rule에 맞춘 사이즈인 asize를 만들어 그것을 oldsize와 비교하게 된다. asize가 작다면 원래 있던 블락을 줄이기 위해 place(ptr,asize)를 호출한다. 이를 통해 줄어든 사이즈가 원래 사이즈보다 16bytes 이상 차이가 난다면 블락을 스플릿한다. 만약 asize가 더 크

다면 asize만큼 mm_malloc으로 할당해 메모리 카피를 진행한뒤 원래 포인터를 mm_free로 할당해제해준다.

-extend_heap

늘릴 힙의 word size를 인자로 받아 그것을 dword에 맞춰지도록 round up한다. 그리고 mem_sbrk를 호출해 bp에 추가된 공간의 base pointer을 저장한뒤, header와 footer 설정 및 epilogue tag도 설정한다. 그리고 coalesce(bp)를 리턴하여 추가된 공간 직전의 block이 free block이라면 합치고 리턴되도록 한다.

-place



받은 free block에 대해 asize만큼의 공간을 사용하게 되는데, 만약 그렇게 채우고 남는 공간이 2*dword=16bytes 이상이면 그것을 따로 하나의 free block으로 나누고 적절히 header와 footer 처리를 해주고 리턴하는 함수이다.

-find_fit

next fit approach를 적용한 free block 탐색 함수이다. 이전까지 탐색하면서 업데이트된 nowpos의 위치부터 리스트의 끝까지 탐색하면서 적절한 free block을 찾으면 그것을 리턴한다. 만약 리스트의 끝까지 발견되지 않았다면 다시 list의 처음부터 기존에 nowpos가 있던 곳(oldpos)까지 탐색해, 그래도 없다면 null을 리턴한다.

2. Global variable and Macros

```
#define WORDSIZE 4
#define DWORDSIZE 8
#define GET(p) (*(unsigned int *)(p))
#define PUT(p, val) (*(unsigned int *)(p) = (val))
#define GET_SIZE(p)  (GET(p) & ~0x7)
#define GET_ALLOC(p) (GET(p) & 0x1)
#define BASE  (1<<13)
#define PACK(size, alloc) ((size) | (alloc))
#define HDRP(bp)       ((char *)(bp) - WORDSIZE)
#define FTRP(bp)       ((char *)(bp) + GET_SIZE(HDRP(bp)) - DWORDSIZE)
#define NEXT_BLKP(bp)  ((char *)(bp) + GET_SIZE(HDRP(bp)))
#define PREV_BLKP(bp)  ((char *)(bp) - GET_SIZE(HDRP(bp) - WORDSIZE))
#define PRV_ALLOC(bp)  (GET_ALLOC(HDRP(PREV_BLKP(bp))))
#define NXT_ALLOC(bp)  (GET_ALLOC(HDRP(NEXT_BLKP(bp))))

static char* imp_list_start = 0;
static char* nowpos;
static int mm_check();
static void* extend_heap(size_t words);
static void place(void* bp, size_t asize);
static void* find_fit(size_t asize);
static void* coalesce(void* bp);
```

WORDSIZE	4byte. header와 footer의 크기가 이 word size에 해당한다
DWORDSIZE	8byte. payload의 시작 주소는 dword size에 aligned되어있다.
GET(p)	p로부터 4byte만큼의 공간에서 unsigned int를 읽는다
PUT(p,val)	위와 같은 공간에 val을 assign한다.
GET_SIZE(p)	p에서 아래 3bit를 비워버리고 size부분만 빼낸다
GET_ALLOC(p)	p에서 lsb를 취해 할당 여부를 얻는다
BASE	extend_heap 시에 확장하는 기본 크기(byte단위)
PACK(A,B)	(A|B)값- size와 alloc bit를 합칠때 사용
HDRP(bp)	bp(payload pointer)의 header를 가리킴
FTRP(bp)	bp(payload pointer)의 footer를 가리킴
NEXT_BLKP(bp)	다음 block의 bp를 가리킴
PREV_BLKP(bp)	이전 block의 bp를 가리킴
PRV_ALLOC(bp)	이전 block의 alloc bit를 검사해 그 여부를 얻는다
NXT_ALLOC(bp)	다음 block의 alloc bit를 검사해 그 여부를 얻는다

char* imp_list_start	implicit list의 시작 지점. mm_init()에서 초기화
char* nowpos	next fit search의 현재 탐색 포인터
mm_check()	heap consistency 검사를 위한 함수
extend_heap()	mem_sbrk() call을 통해 heap을 최소 BASE만큼 늘릴 함수
place()	받은 free block bp에 대해 tag를 최신화 하고 만약 크기가 남을 시 추가적으로 split까지 수행하는 함수
find_fit()	list에서 할당이 가능한 free block이 있는지 탐색해 리턴할 함수
coalesce()	이전 또는 다음 block이 free block 이라면 coalesce를 수행할 함수

3. mm_check()

heap consistency를 검사하는 함수이다. 포인터 p가 implicit list의 시작 지점인 imp_list_start부터 시작해 계속 다음 블록으로 움직이며 결국 블록의 사이즈가 0인(epilogue footer=end of list) 곳에 도달할때까지 루프를 돈다. 여기서 블록의 사이즈가 8로 나누어떨어지지 않으면 dword alignment가 지켜지지 않았음을 출력하고, 또 header와 footer가 일치하지 않는다는 것은 특별히 초기화에 문제가 없다고 가정하면 payload가 겹쳤을 수가 있고 이를 출력해주게 된다. 문제가 있으면 0, 문제가 없다면 1을 리턴하는 함수이다.