**CSED211 컴퓨터SW시스템개론**

**Lab Assignment #7: Malloc Lab:**

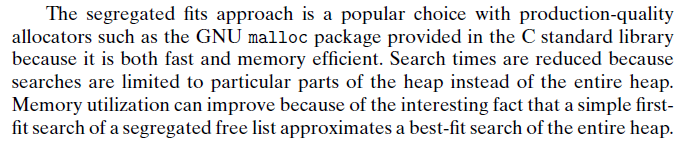
**Writing a Dynamic Storage Allocator**

**20180551**

**컴퓨터공학과**

**이준석**

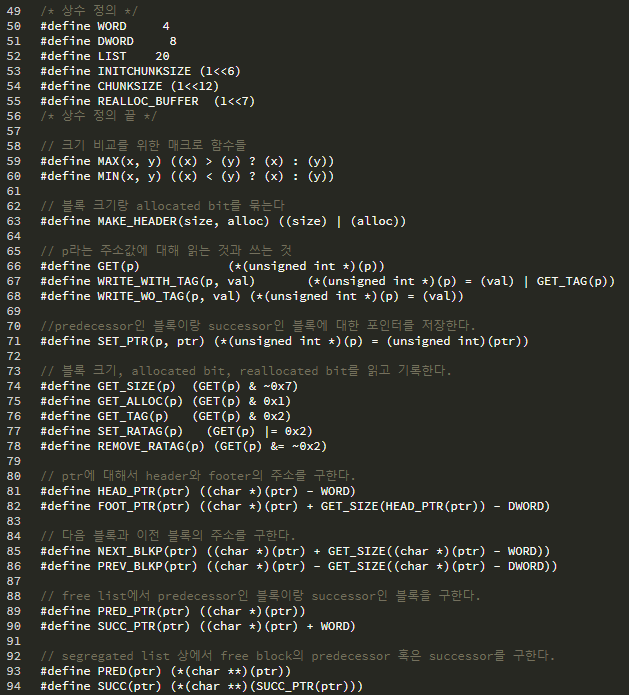
1. **구성**

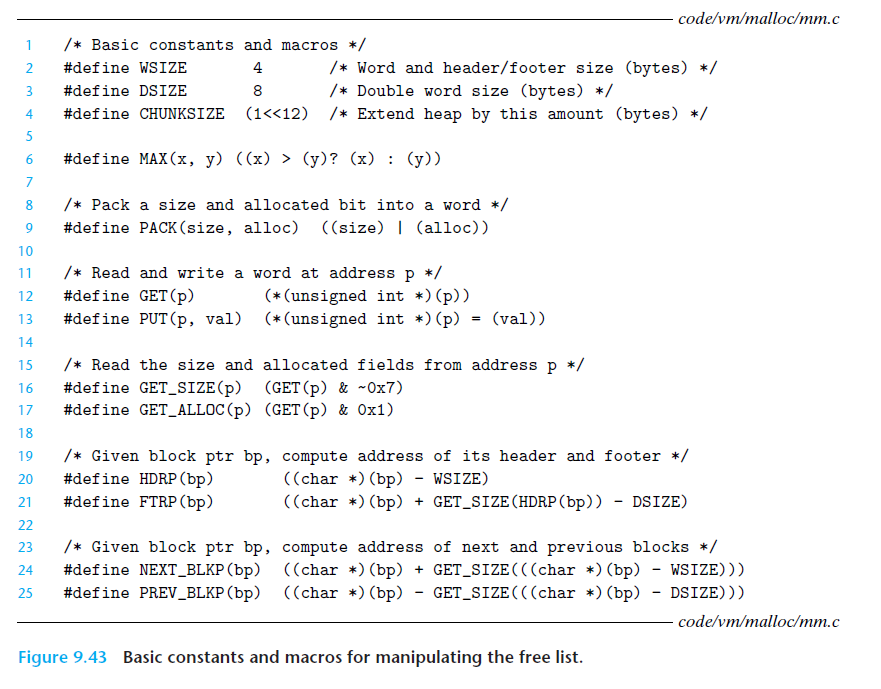


책을 보면 위의 그림처럼 GNU malloc 또한 segregated free list를 이용한다는 것을 알 수 있다. 게다가 평가 방식을 살펴보면 malloc package를 이용한다. 그래서 본인도 segregated free list를 이용하여 구현하였다.

그러고나서 각각의 free list들을 implicit free list들로 구성하였다.

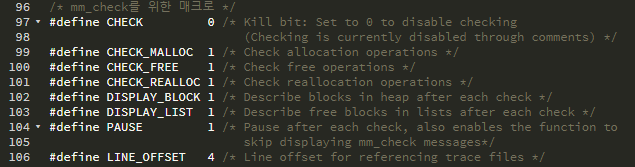
1. **구현**
   1. **상수 정의 및 매크로 함수들**



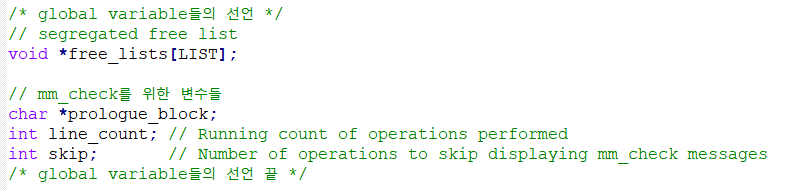


교재 893쪽을 참고하여 작성하였다. 위의 그림은 교재의 893쪽에 나와 있는 그림이다.

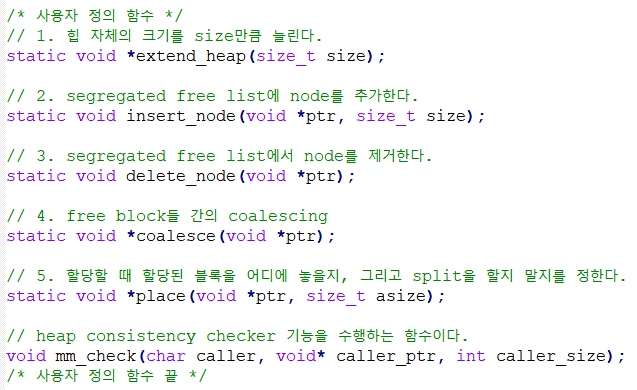
* 1. **mm\_check(heap consistency checker)를 위한 상수 정의**



* 1. **전역 변수**

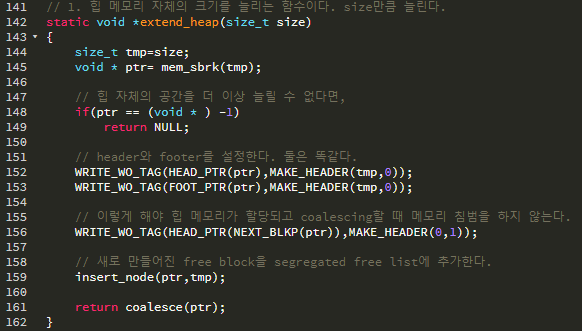


* 1. **사용자 정의 함수**

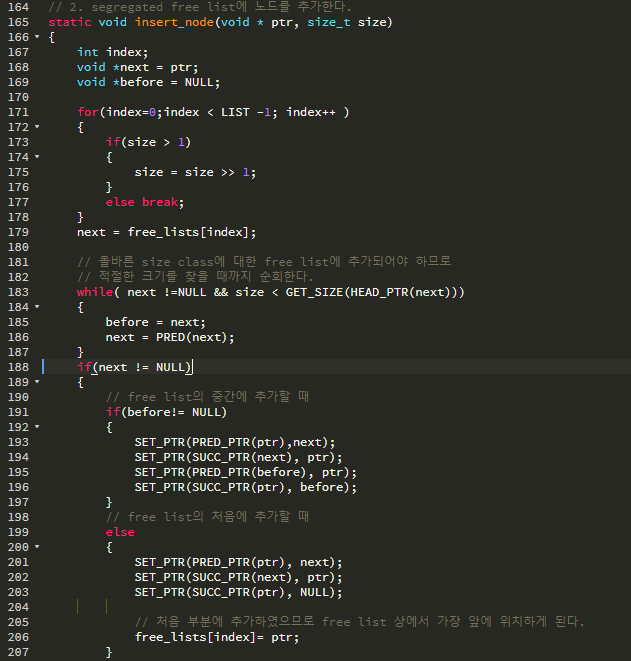


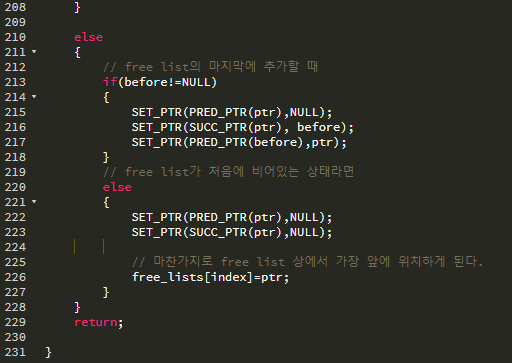
각 함수에 대한 자세한 설명은 각 함수의 정의 부분에 주석으로 달려있다.

* + 1. **extend\_heap**

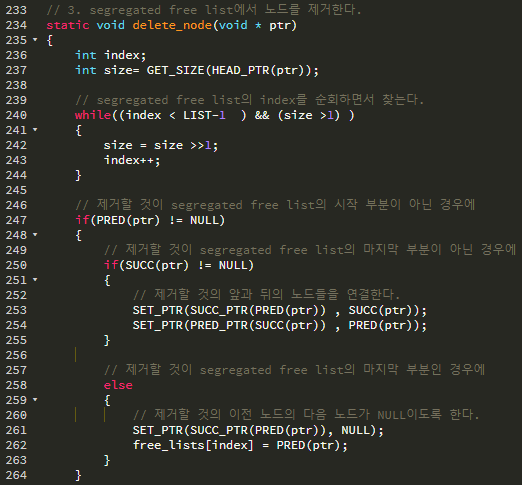


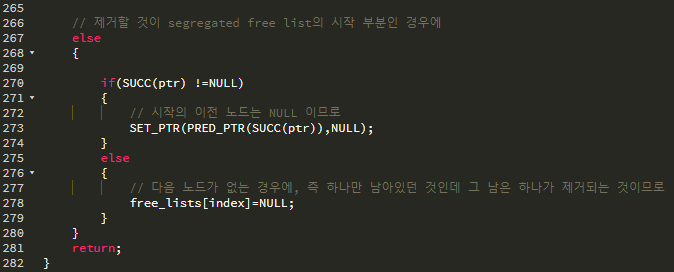
* + 1. **insert\_node**



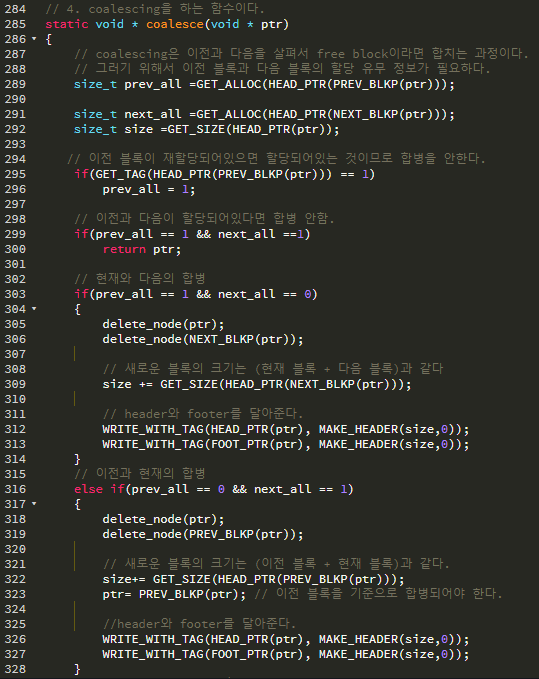


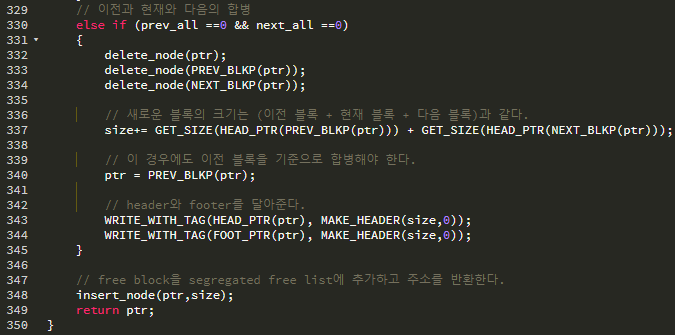
* + 1. **delete\_node**



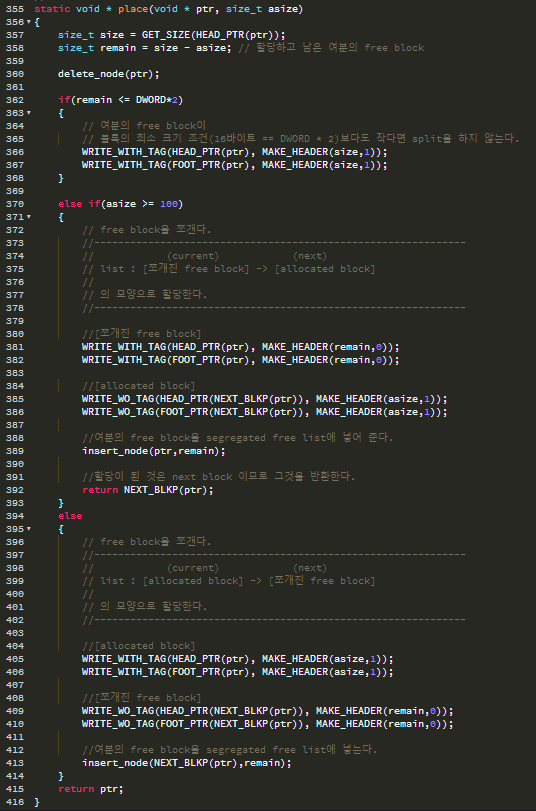


* + 1. **coalesce**



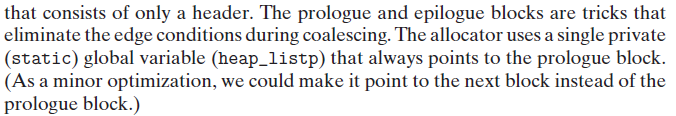
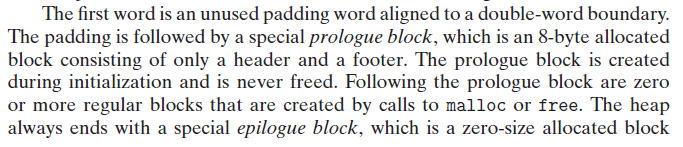


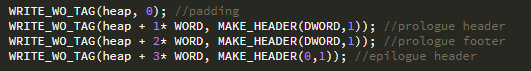
* + 1. **place**



* 1. **mm\_init / mm\_malloc / mm\_free / mm\_realloc**

1. mm\_init

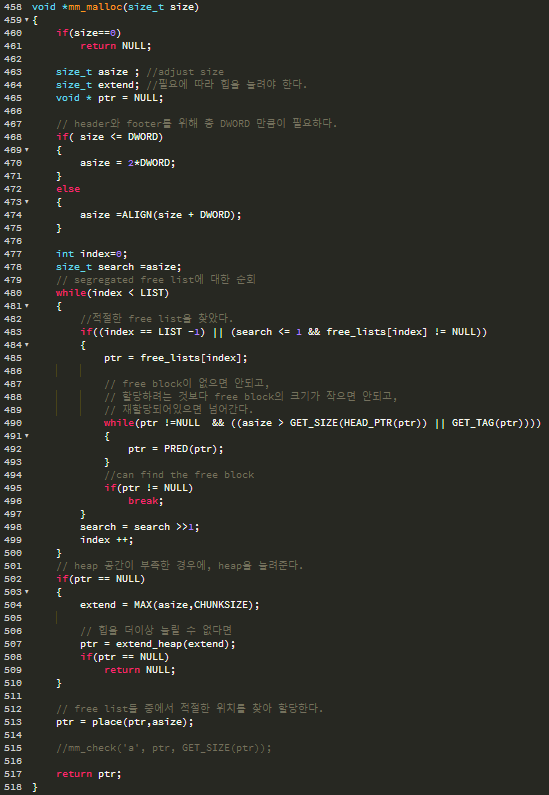
교재(891쪽 하단 ~ 892쪽 상단)를 보면 위의 그림처럼 작성되어 있다. 즉, padding, prologue header와 footer, eplilogue header가 필요하다는 것을 알 수 있다.



그래서 mm\_init 상에서 위의 그림과 같은 부분이 등장한다.

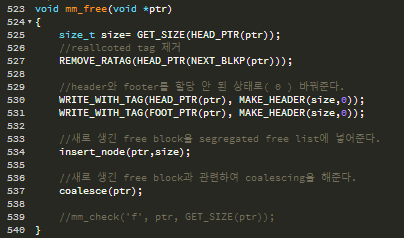
1. mm\_malloc

인자로 넘겨준 size값에 대해 alignment requirement를 충족시켜주고 적절한 free block을 찾아서 할당한다.



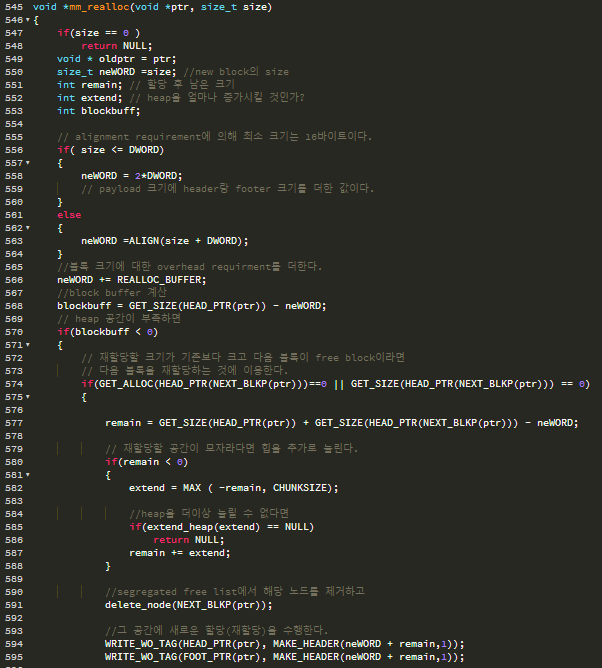
1. mm\_free

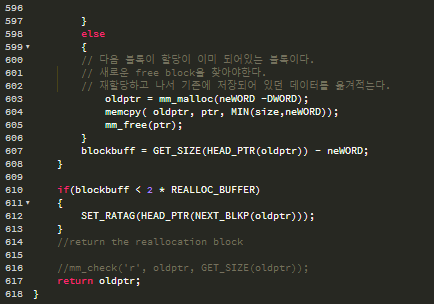
allocated 관련 bit들을 0으로 만들고, 그 블록을 free list에 추가한다. 그리고 coalescing을 수행한다.



1. mm\_realloc

단순히 malloc과 free를 이용하여 구현하지는 않았다. 만약 할당된 것을 재할당할 때 할당된 것 앞 혹은 뒤에 free block이 있다면 그것을 이용하여 재할당을 하였다.

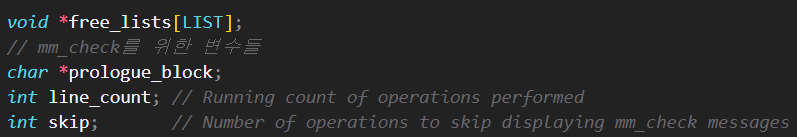




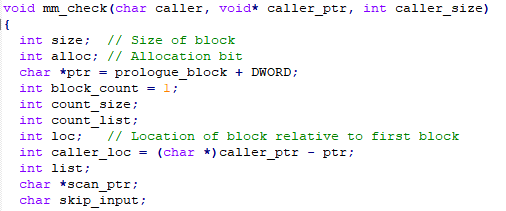
* 1. **mm\_check**

mm\_malloc을 하면 size만큼 heap을 할당한 이후에 그것의 위치를 가리키는 포인터를 반환한다. 그 포인터를 이용하여 그 부분이 제대로 구성되었는지를 확인한다. (물론, free 혹은 realloc도 포인터와 관련있으므로 그 포인터를 이용할 수 있다.)

* mm.c 파일 하단에 주석으로 처리 되어 있다.



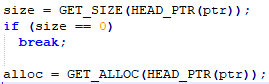
mm\_check에서 이용되는 전역 변수들이다.



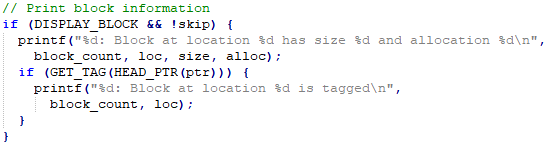
caller는 mm\_malloc에서 호출되었는지, mm\_free에서 호출되었는지, mm\_realloc에서 호출되었는지 등을 구별해주기 위해 문자를 넣어주는 부분이다.

caller\_ptr은 앞서 언급했던 것처럼 malloc의 반환 값 등이다.

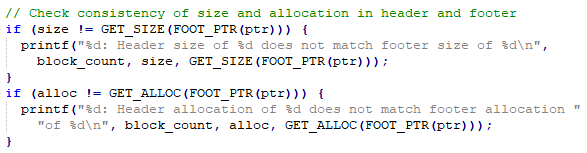
caller\_size는 caller\_ptr에 대응되는 block size이다.



size bit들과 allocated bit를 받아서

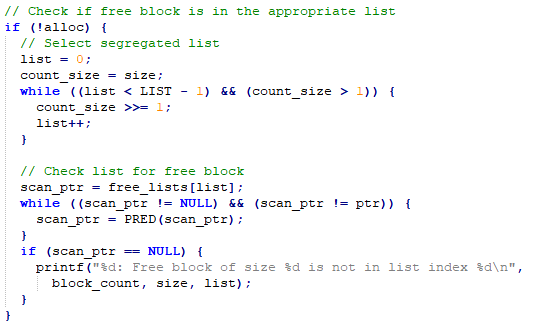


블록의 정보를 출력해주고



header로부터 받은 정보와 footer의 정보가 일치하지 않는다면 오류 메시지를 띄워준다.

(header와 footer는 같아야 한다.)



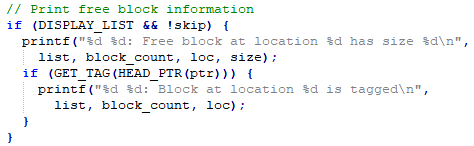
free block인 경우에, segregated free list에서 적절히 존재하고 있는지를 확인한다.



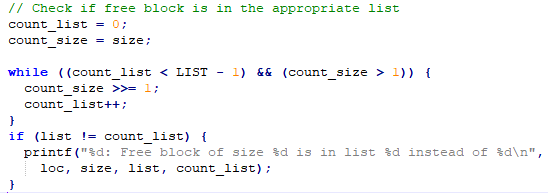
다음 블록으로 넘어간다.



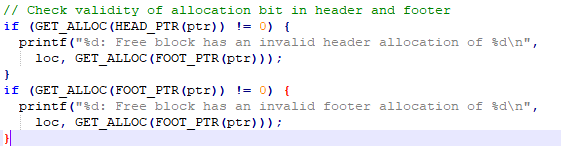
segregated free list 상에 있는 모든 free list에서 확인한다.



블록의 정보를 출력하고



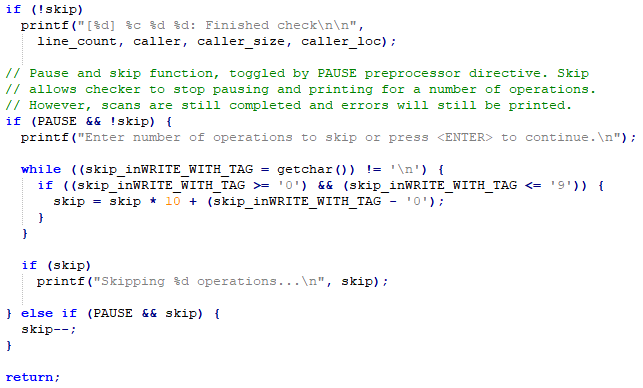
free block이 적절한 free list 안에 들어 있는지 확인한다.



free block인데 header 혹은 footer에서 allocated bit가 0이 아니면 틀린 것이다.



다음으로 넘어가서 반복한다.



pause할 수 있고 코드를 약간씩 수정하여 필요가 없는 테스트들은 건너 뛸 수 있다.