**实验报告**

**刘雅迪**

**计26**

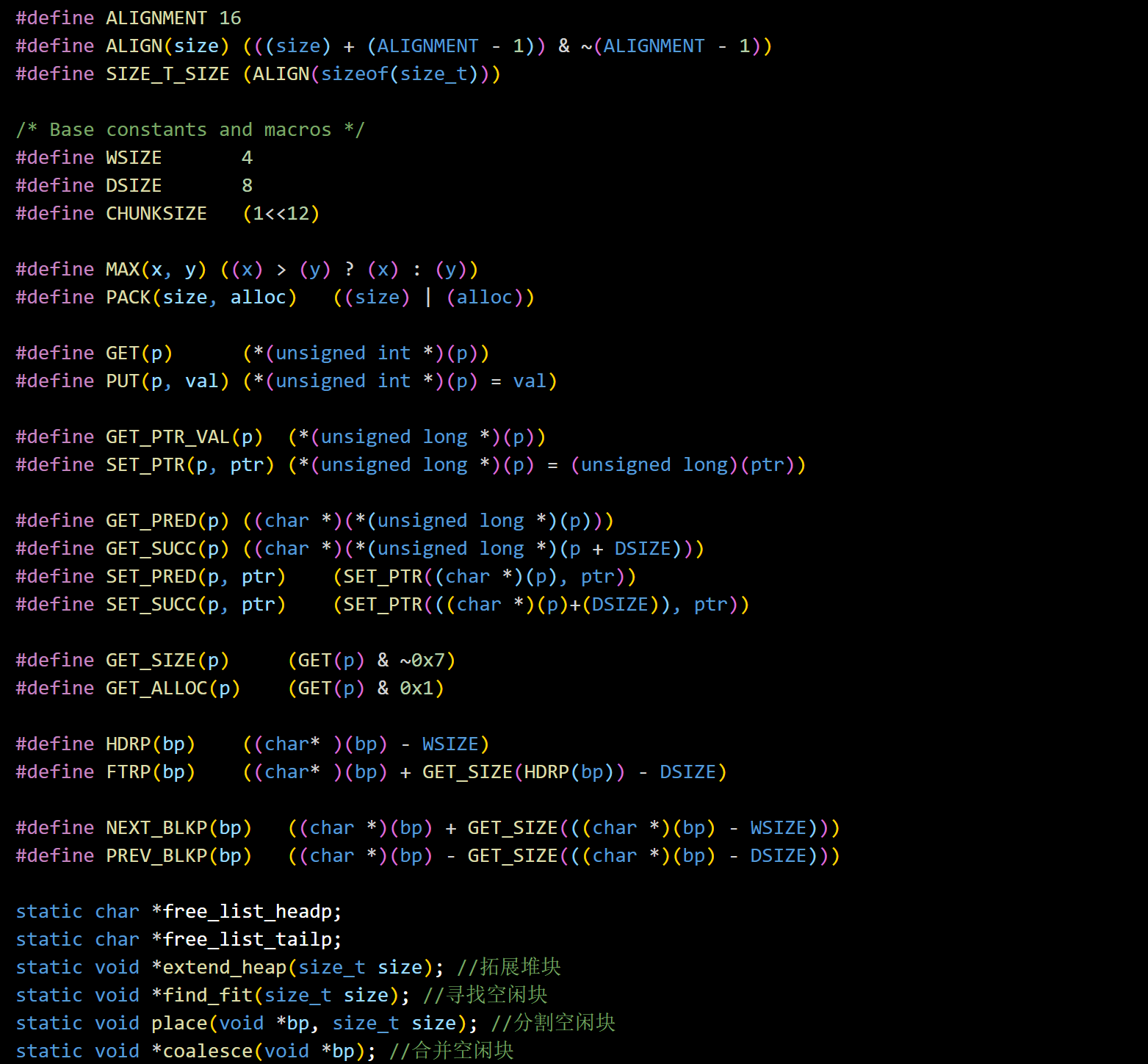
一、实验构思：

使用显示空闲链实现动态内存分配器。将堆组织成一个双向的空闲链表，在每个空闲块中，都包含一个pred和succ指针。这样首次适配的分配时间从块总数的线性时间减少到了空闲块数量的线性时间。

双向链表包含一个free\_list\_headp和free\_list\_tailp节点指针，以便对空闲链表进行插入修改等操作。

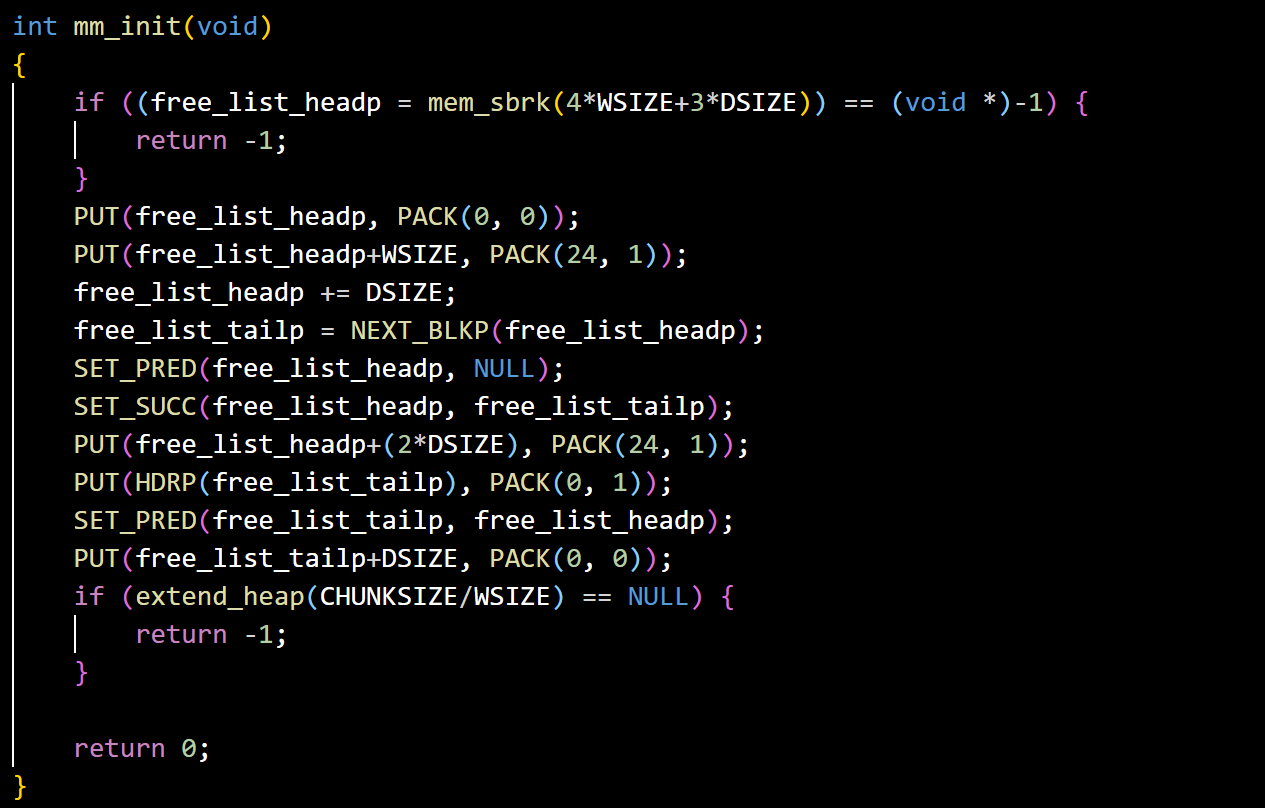
二、实验具体内容实现：

1. 定义的宏与函数：



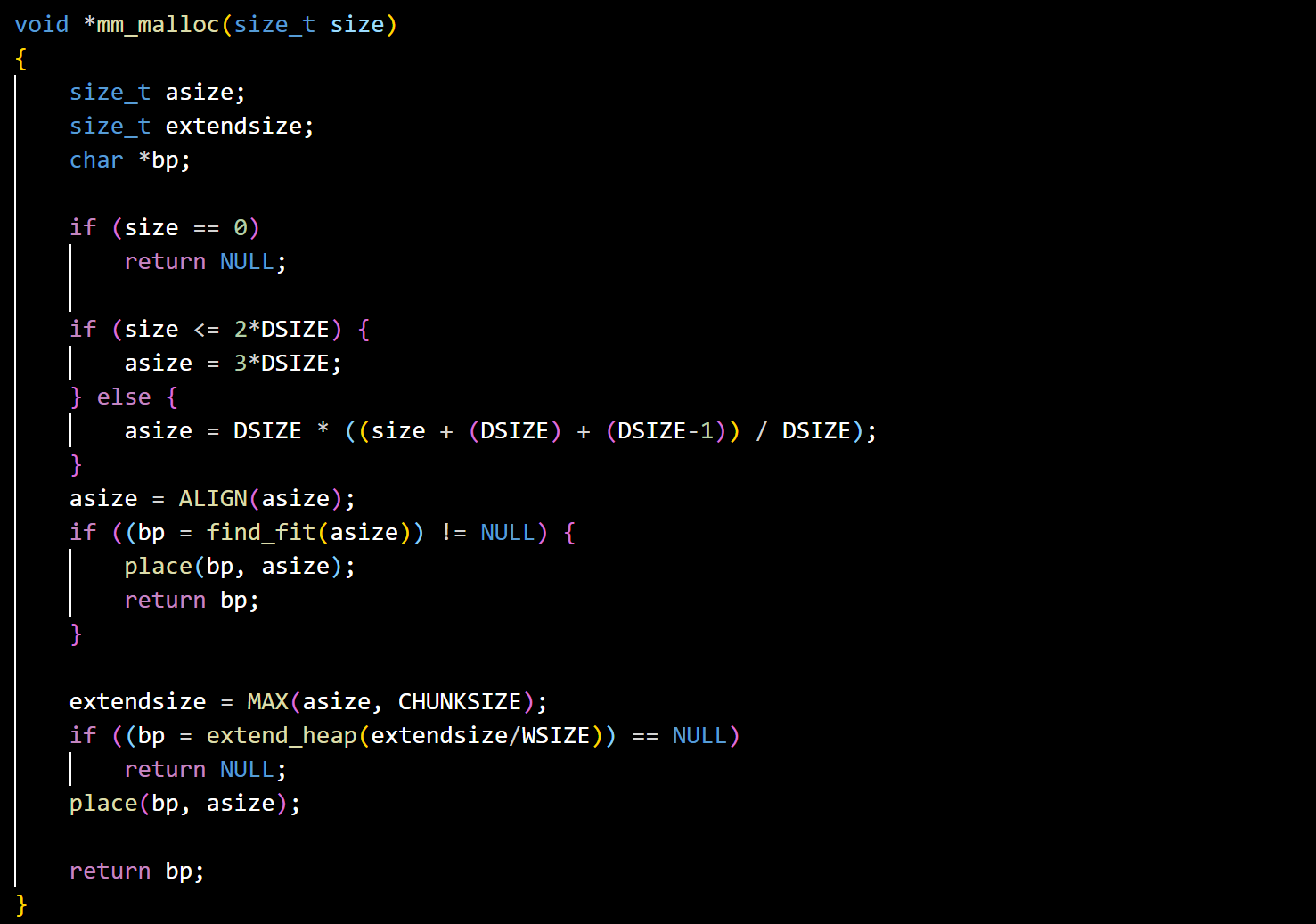
1. int mm\_init(void)

显式空闲链表初始化时分配一个空的堆区，头部和尾部都要添加一个4字节的Padding，以满足对齐要求。



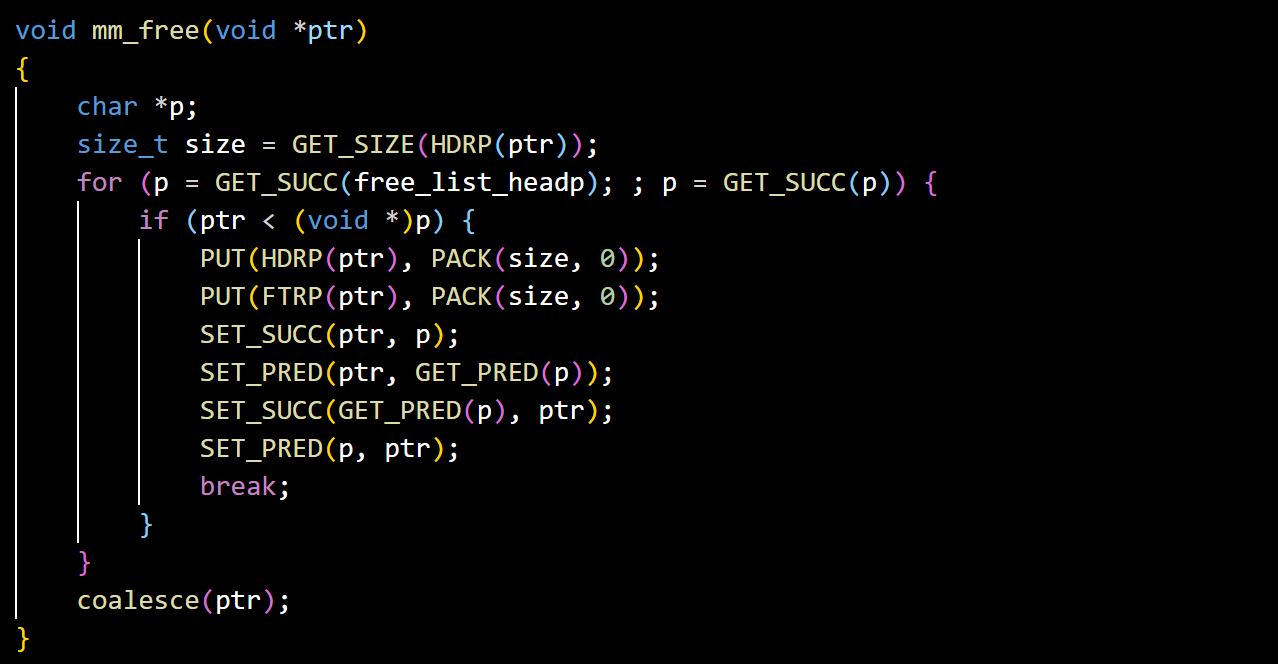
1. void \*mm\_malloc(size\_t size);

设定asize，如果size<16bytes，那么用最小块大小24bytes代替；否则，需要只考虑头部尾部的开销，还需要进行对齐。然后在空闲链表使用首次适配的方式，使用find\_fit函数进行顺序搜索，找到第一个适配的空闲块后运用place函数放置。



1. void mm\_free(void \*ptr);

在空闲链表中顺序查找，找到第一个空闲的块，然后设置相应的PRED和SUCC的值。释放掉这个块后调用coalesce对这个块进行合并，操作在常数时间内。



1. void \*mm\_realloc(void \*ptr, size\_t size);

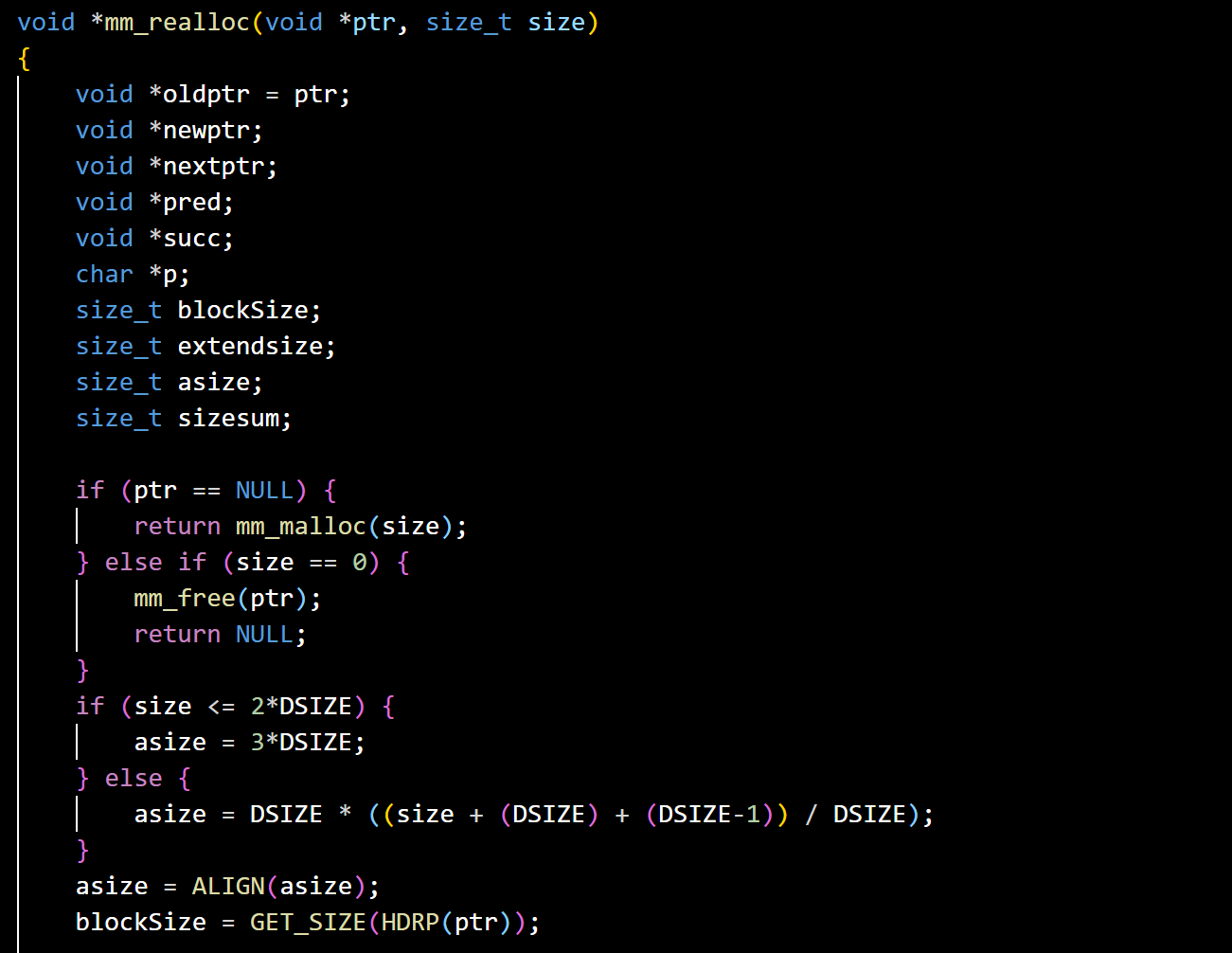
首先对ptr和size进行判断：如果ptr==NULL，调用mm\_malloc(size)；否则如果size==NULL，调用mm\_free(ptr)。

进行到后面的代码时ptr!=NULL并且size>0，首先对size进行调整，添加头部尾部等，然后判断asize和这个块的大小关系：

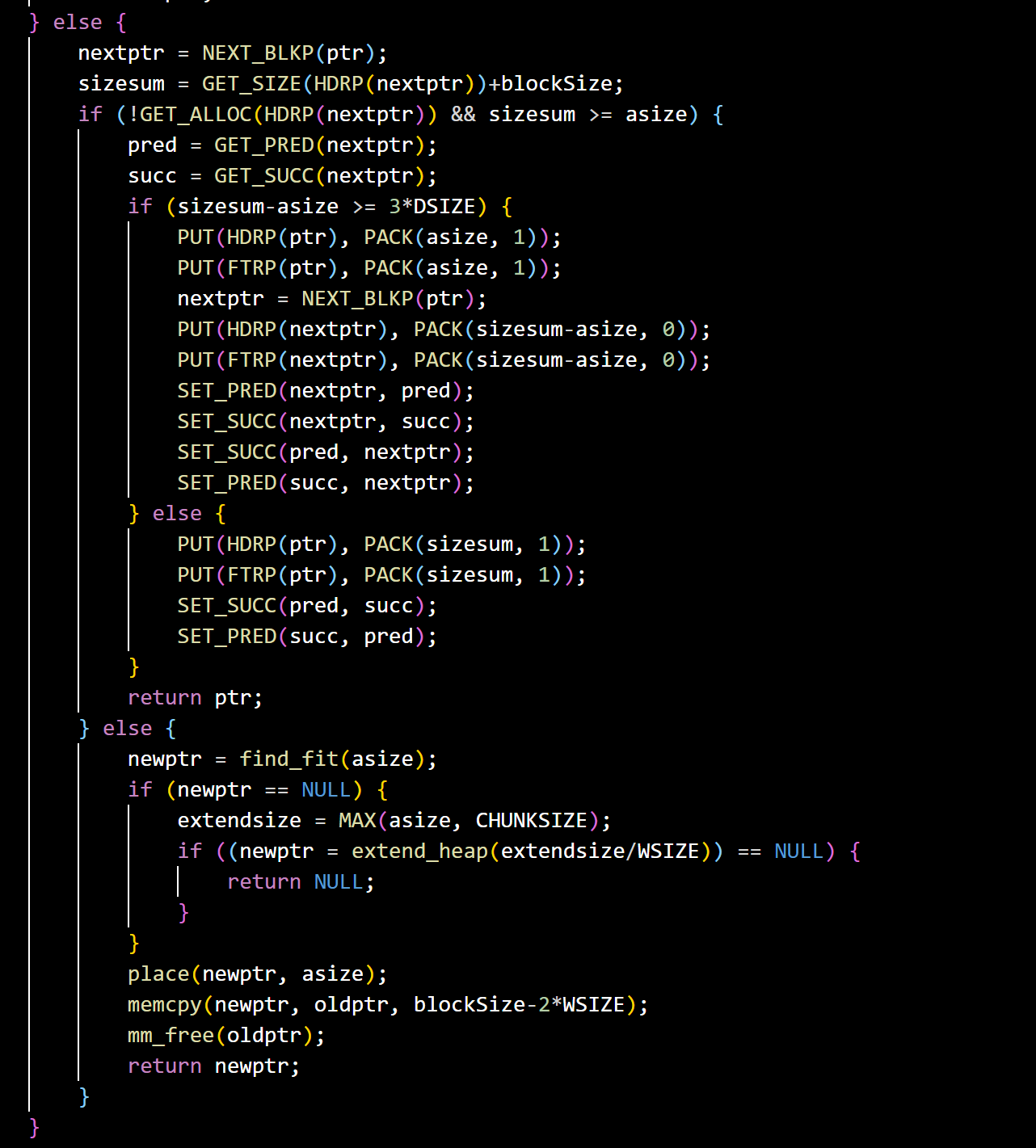
·如果asize == blockSize，什么都不做；

·如果asize < blockSize，与place函数操作类似，但是不直接调用place函数；

·如果asize > blockSize，考虑下一个块是否已分配并且考虑其大小。







三、实验感想：

这次实验对我来说很难，感觉是三个实验中最难的一个。程序可以运行的瞬间无疑是开心的，但是分数不高，尝试改了好久都不知道为什么吞吐率那么低，怎么改都没什么用。无奈只能放弃了，尽管性能分可能没那么高，但至少还有一些基础分数。希望助教哥哥or姐姐能手下留情，多给我点分数，为这门课画上一个完美一点的句号。

四、参考资料：

1. 《深入理解计算机系统》第九章
2. 《CSAPP(CMU 15-213)：Lab6 Malloclab详解》，

https://blog.csdn.net/qq\_42241839/article/details/123697377