关于第三次实验的一些更细致建议与时间安排（仅供参考）

1、总的数据结构方案选择

大致可以分为隐式空闲链表（如 ppt 里的）以及显式空闲链表两类，进一步的可以增加 Segregated 支持，即分为多个链表，每个链表所支持的数据块的大小不同（比如第 n 个链表支持的数据块大小在[2^n，2^{n+1}）间）。上述链表建议均为双向链表。

明显地，按照隐式、显式、Segregated 显式这一顺序，查询的复杂度逐步降低，吞吐率也会越高，这已经为往年的实验评分所证实；一个可能的优化是每个链表中均以大小非降序维护所有空闲块。

当然，具体采用什么实现策略，要看同学们的具体情况。往年有的高分同学是按照隐式、显式、Segregated 显式这一顺序递进实现，也有的一步或者二步到位。不管如何，着手时要尽快确定适合自己的方案。

2、辅助函数（宏）：

注：下列函数并非为框架提供，而是作为同学们完成实验过程中，设计和编写辅助函数/宏的参考；

1) find_fit(size_t size)： 在空闲链表中按 size 查找不小于 size 的空闲块。
2) freelist_insert(void* p)：将一个空闲块插入空闲链表。
3) freelist_remove(void* p)：将一个空闲块移除出空闲链表。
4) coalesce(void* p)：合并该与某空闲块地址上相邻的空闲块，合并后转为一个空闲块。
5) place(void* p， size_t size)：把空闲块分配给需求 size，如果空闲块较大（注意对齐）可以分割出新的空闲块避免内部碎片。

6) extend_heap(size_t size)： 扩展堆。

7) {get， put}_ptr：读/写 64 位的指针。

其它还有一些更为基础的对于链表的基本操作，比如转至当前块 p 的 header、块的大小、至上一个块（物理意义）、至下一个块（物理意义）等等；这个依赖于具体实现。

3、性能调优的一些推荐策略

1) 采用 Segregated 显式空闲链表方式，使得查找适合大小的自由块更加高效，这提高了分配和释放操作的速度。

2) 在释放内存时及时进行相邻自由块的合并，这有助于减小外部碎片，提高内存利用效率，对于长时间运行的程序和频繁进行内存分配和释放的情况具有良好的性能。

3) 完善 mm_realloc 的逻辑，而不是简单的释放再分配。可以尝试将旧块通过与相邻的空闲块合并的方式就地分配新的空间，同时复制原有内容。该方法避免了不必要的重新寻找空闲块甚至开辟新的堆空间，即提高了空间利用率，又提高了运行效率。

4) 针对具体测试 trace 的优化，如先观察 binary-bal、realloc2-bal 等的操作规律，再采取措施，虽然有些 tricky，但是是允许的。

最后，如果采用二叉搜索树等更高级的空闲链表设计形式，能够进一步提高内存的利用率，同时将适配搜索的时间复杂度降低，获得更好的性能表现。

4、推荐的实验流程和时间分配

建议大家尽早开始实验

2 周时间熟悉实验框架，了解实验要求，完成初版符合正确性要求的代码。

2 周的时间尝试，选择和确定最终实现的算法。

2 周的时间针对性地进行性能调优，包括常数上的优化以及针对测例或者算法上的优化。