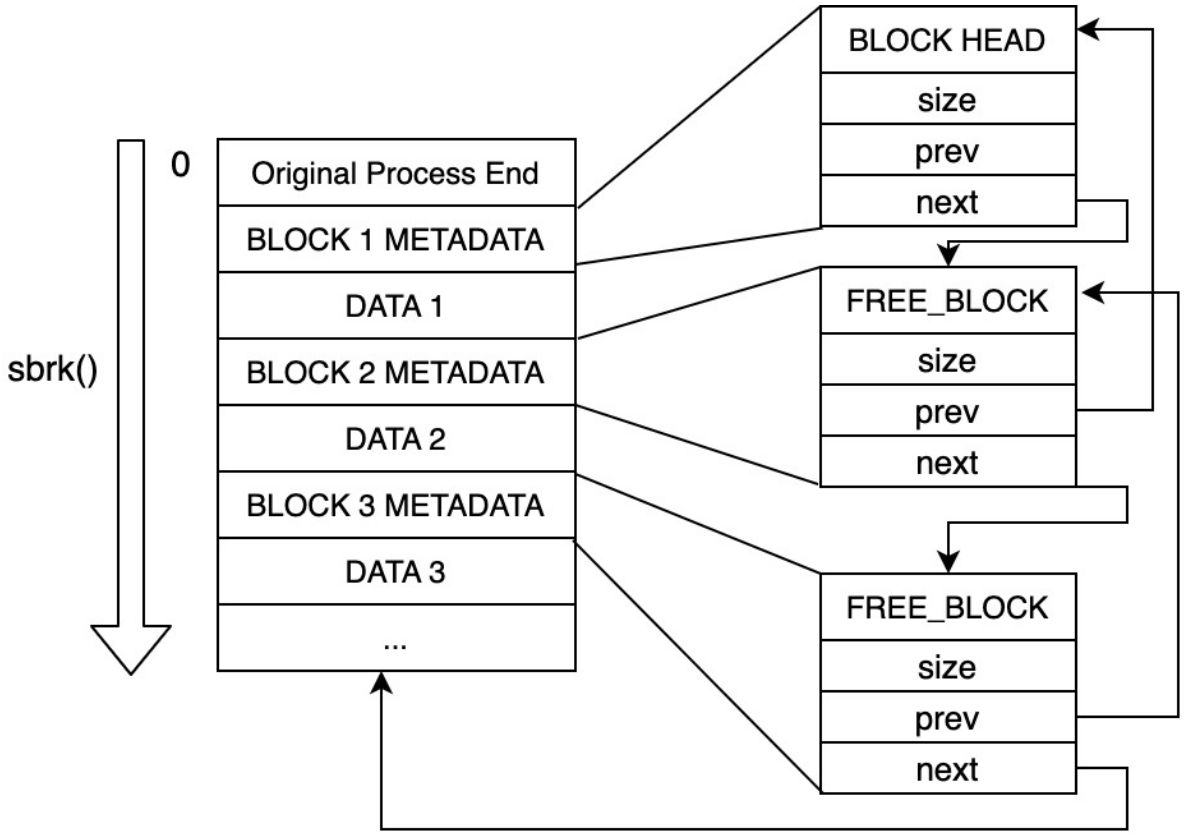
Report of Homework 1

Name: Chenglong Tang, NetID: ct265

- Target: I am asked to implement my own version of 2 memory allocation functions from the C standard: malloc() and free(). In this assignment, system call sbrk() is used to allocate additional space in heap, which is returned to users.



- Implementation Description: As you can see above. In this assignment, a free block doubly linked list is maintained to store available blocks that users can get. Data block resides closely after each free block for user to store data. When I use malloc() function, it searches through the linked list to find the block meeting the size requirements. If nothing found, sbrk() is called to allocated additional memories. While free() function is called, it emplaces the returned chunk of memory back to the free list in address ascending order. Then a merge happens when the inserted blocks could be merged with blocks nearby(previous and next).

ff means “First Fit”. When it’s called, it examines free list and allocate an address from the first free region with enough space that fits required allocation size.

bf means “Best Fit”. Different from First Fit, Best Fit examines all the free list blocks and find the minimum number of bytes that’s greater than or equal to the requested allocation size.

- Performance Result Presentation:

For First Fit, the result is:

```
ct265@vcm-23181:~/my_malloc/alloc_policy_tests$ ./small_range_rand_allocs
data_segment_size = 3754480, data_segment_free_space = 323504
Execution Time = 13.340473 seconds
Fragmentation  = 0.086165
ct265@vcm-23181:~/my_malloc/alloc_policy_tests$ ./equal_size_allocs
Execution Time = 15.675088 seconds
Fragmentation  = 0.450000
ct265@vcm-23181:~/my_malloc/alloc_policy_tests$ ./large_range_rand_allocs
Execution Time = 45.550756 seconds
Fragmentation  = 0.102378
```

For Best Fit, the result is:

```
ct265@vcm-23181:~/my_malloc/alloc_policy_tests$ ./small_range_rand_allocs
data_segment_size = 3534544, data_segment_free_space = 101592
Execution Time = 4.757582 seconds
Fragmentation  = 0.028743
ct265@vcm-23181:~/my_malloc/alloc_policy_tests$ ./equal_size_allocs
Execution Time = 15.531942 seconds
Fragmentation  = 0.450000
ct265@vcm-23181:~/my_malloc/alloc_policy_tests$ ./large_range_rand_allocs
Execution Time = 66.182354 seconds
Fragmentation  = 0.044918
```

- Result Analysis:

For **equal_size_allocs**, it is obvious that two methods have exactly **same** performance on Speed and Fragmentation. Though Best Fit normally has to traverse the whole list in order to find the best block, it does not have to do so when it has already found the block that has exactly same size as requested.

For Fragmentation: **Best Fit** has **lower fragmentation** value than that of **First Fit** in both **small_range_rand_allocs** and **large_range_rand_allocs**. In the test case, Best Fit always tries to find the smallest chunk whose size is greater than requested size, thus leaving relatively small part in the free list. It causes the size of "free list" to be smaller in Best Fit method. So the fragmentation value is lower.

For Speed: **First Fit** has **higher speed** than that of **Best Fit** in **large_range_rand_allocs** but has **lower speed** than Best Fit in **small_range_rand_allocs**. For the testcase that deals with large range random allocation, First Fit is more likely to divide chunks into more pieces, thus increasing the size of freed list. So First Fit has more overhead in finding the requested block in large_range_rand_allocs. However, when it comes to allocation of memory pieces with small size range, First Fit is quicker since those pieces have similar size and picking the first one, the last one, or the best one has no difference. In this case, picking the first one will significantly reduce the runtime because you don't need to traverse the whole freed list.

- Recommendation: There is no such method as the best in this assignment. Best Fit plays better in large chunks allocation but worse in small size allocation than First Fit. It still depends on the environment to predict which one is better.