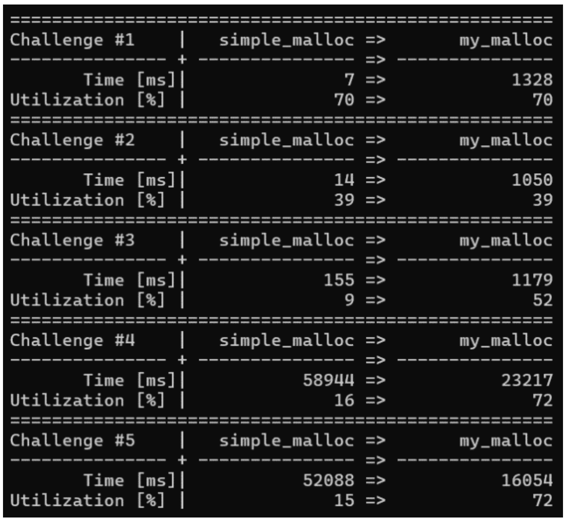
Malloc Challenge

Naho Yokoyama

Implementing best-fit

Instead of choosing first slot (that has its size greater than or equals to the size we are looking for), I modified the program so that it chooses the slot that has the size closest to the size we are looking for (of course, its value should be greater, not less). The result are as follows, and it is clear that the utilization increased but since we have to go through all slots, the time increased a lot.



```
====================================================
Challenge #1    |   simple_malloc =>       my_malloc
--------------- + --------------- => ---------------
       Time [ms]|               7 =>            1328
Utilization [%] |              70 =>              70
====================================================
Challenge #2    |   simple_malloc =>       my_malloc
--------------- + --------------- => ---------------
       Time [ms]|              14 =>            1050
Utilization [%] |              39 =>              39
====================================================
Challenge #3    |   simple_malloc =>       my_malloc
--------------- + --------------- => ---------------
       Time [ms]|             155 =>            1179
Utilization [%] |               9 =>              52
====================================================
Challenge #4    |   simple_malloc =>       my_malloc
--------------- + --------------- => ---------------
       Time [ms]|           58944 =>           23217
Utilization [%] |              16 =>              72
====================================================
Challenge #5    |   simple_malloc =>       my_malloc
--------------- + --------------- => ---------------
       Time [ms]|           52088 =>           16054
Utilization [%] |              15 =>              72
```

Implementing free list bin

Since the time of the best-fit version was long, I implemented free list bin to