


A Dynamic Memory Allocation Analogy

 coursera.org/learn/interacting-system-managing-memory/supplement/kcCY7/a-dynamic-memory-allocation-analogy

malloc and **free** can be confusing to novice programmers, but they are crucial to producing correct, efficient, and clean code. To help you understand them, consider the following analogy: You are at the bus depot and there are 50 lockers available for rent. To request a locker, you simply tell the worker at the window how many lockers you need. You will be given a contiguous stretch of lockers, and you will be told which is the first locker that is yours. For example, you might say "5 lockers, please," and you might be told "Your starting locker is number 37," at which point, you have been given lockers 37 through 41. You might also be told "No, sorry." because there are not 5 contiguous lockers available for rent. In response to your successful request, the worker at the window records that 5 lockers are in use, starting at locker 37.

When you are done using your lockers, you return to the window and tell the worker "I'm done with the lockers starting at 37," and those 5 lockers now become available to someone else behind you in line. When you return your lockers, you free all or none of them. You can't return a subset of your request. Also, you need to keep track of your starting locker (37), because that is how the worker has recorded your booking. In technical terms, you may call **free** only with a pointer that was returned by **malloc**. Furthermore, you cannot return your lockers twice. Even if you and your partner are both using the lockers, only one of you should return them. Again, in technical speak, you may call **free** only once, even if there are multiple pointers to that location in memory. Freeing the same location in memory multiple times is illegal.



Completed
