Unit 14: Dynamic Storage Allocation

Using dynamic storage allocation, a program can obtain blocks of memory as needed during execution. The can mean allocating a single block of memory, reallocating a block of memory for changing its size, or allocating parts of a large data structure piece by piece. Dynamically allocated memory must also be deallocated by calling the free function, before the program terminates. This is different to variables and arrays that we used up to now - dynamically allocated storage requires (in C) that the programmer also takes care of deallocating that storage properly.
Relevant for this unit are the following Book Sections:

- Section 17.1: Basic dynamic storage allocation
- Section 17.2: Dynamically allocated strings, which provide more flexibility than ordinary character arrays.
- Section 17.3: Dynamic storage allocation for arrays in general
- Section 17.4: Storage deallocation – releasing blocks of dynamically allocated memory when they are no longer needed

Project u14: Dynamic Inventory - Modification 2

Modify the inventory.c program (or your own project) so that the `inventory` array is allocated dynamically and later reallocated when it fills up. Thus, your the modified version of inventory only runs out of memory if no memory is available to your program anymore, it is no longer bound by some fixed size of the array. Use `malloc` initially to allocate enough space for an array of 8 part structures. When the array has no more room for new parts, use `realloc` to double its size. Repeat the doubling step each time the array becomes full. Make sure you also deallocate (free) the allocated memory.
Refer to the provided test-cases and compare the output of your program with the provided output of the test-cases. Your program should behave exactly like the test-cases before you hand it in. Hand in your program as `u14_invmod2.c`.