Malloc and Free Simulation

John Ferrer
Ava Shaw

Usage: ./mymalloc

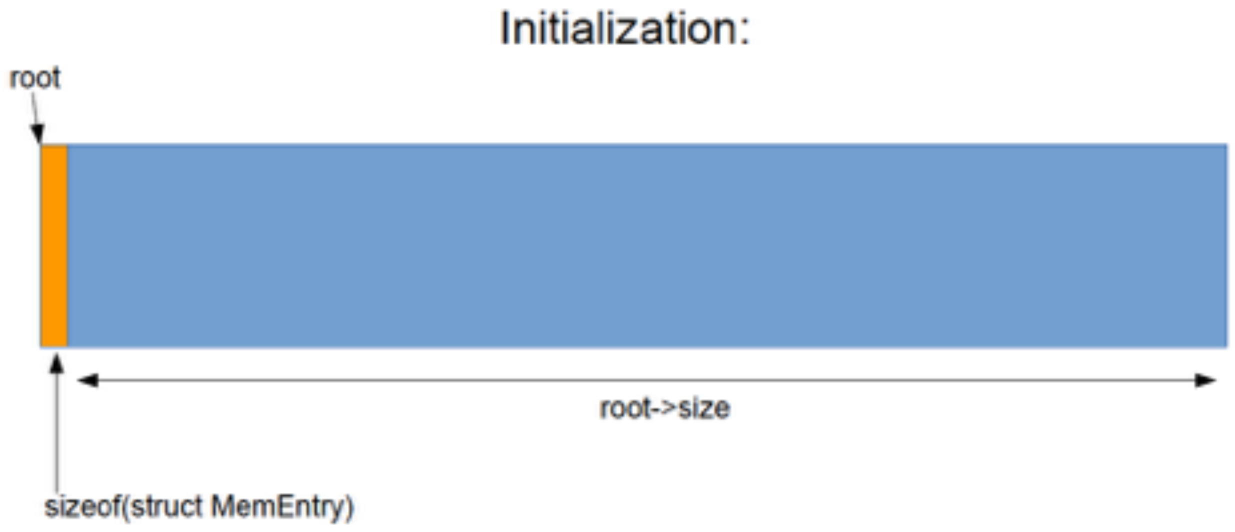
Description: This program is a simulation of *improved* malloc() and free() library calls by detecting errors such as:

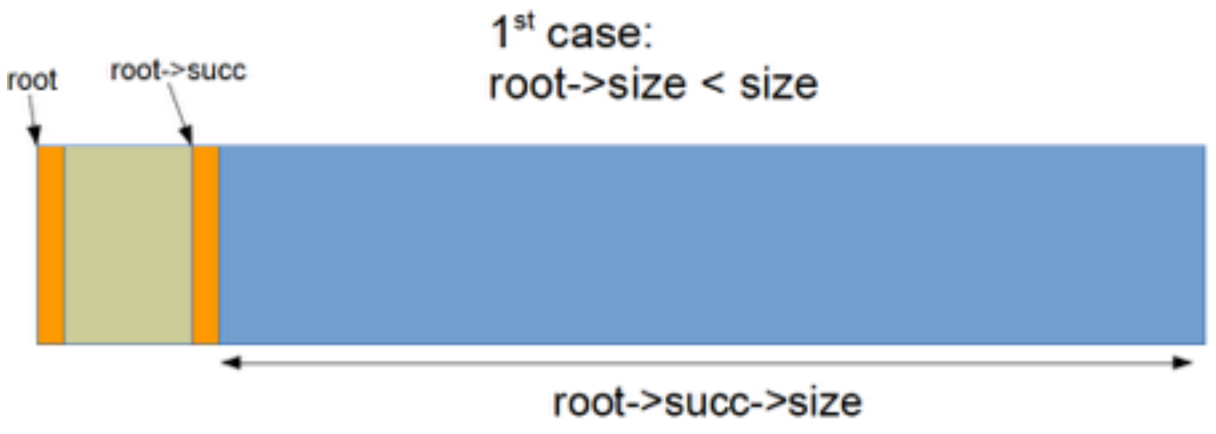
- Allocating 0 bytes of memory
- Allocating a block when you are out of memory
- Freeing a null pointer
- Dynamically moves memory entries to deal with fragmentation
- Freeing pointers to an address which does not have a valid malloc entry meaning malloc was not called
- Freeing pointers which were not allocated

Structure:
Our simulation uses a static char array to hold the memory entry structures and an array of void pointers to the blocks containing the memory entries.

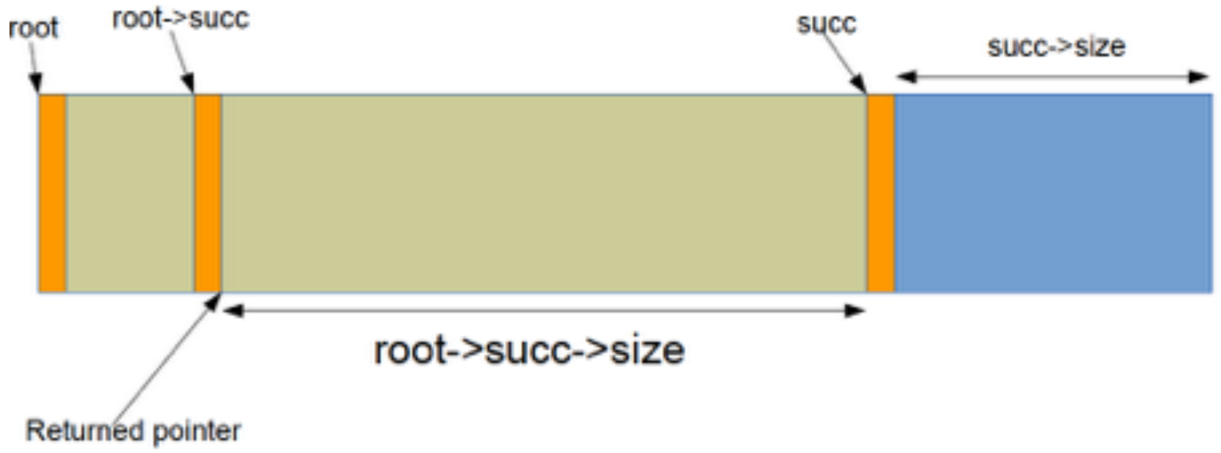
myMalloc:They are all initialized at NULL to keep track of the validity of the memory entires (i.e. changing the flag when it is legitimately malloced) Every time a new entry is added a new header for the next memory entry is created to be filled when the next item is added, this way we can dynamically arrange the entires by knowing how much space has been used or is available. Operates at an O(n) efficiency.

myFree: The pointer passed in is check for validity (that the flag was set that it was malloced()); subtracts the size of the memory entry. The block is freed and the surrounding blocks are merged to not allow for fragmentation. Operates at an O(n) efficiency.

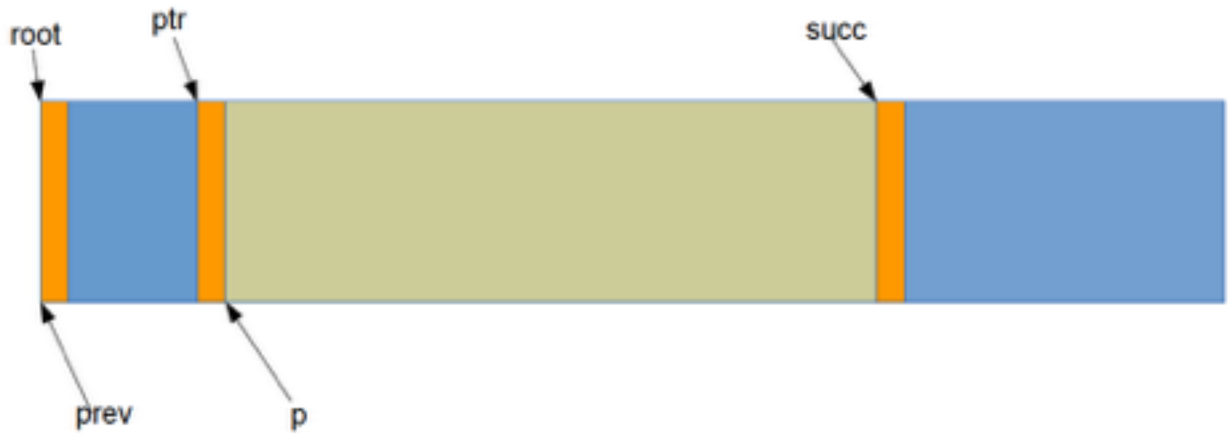




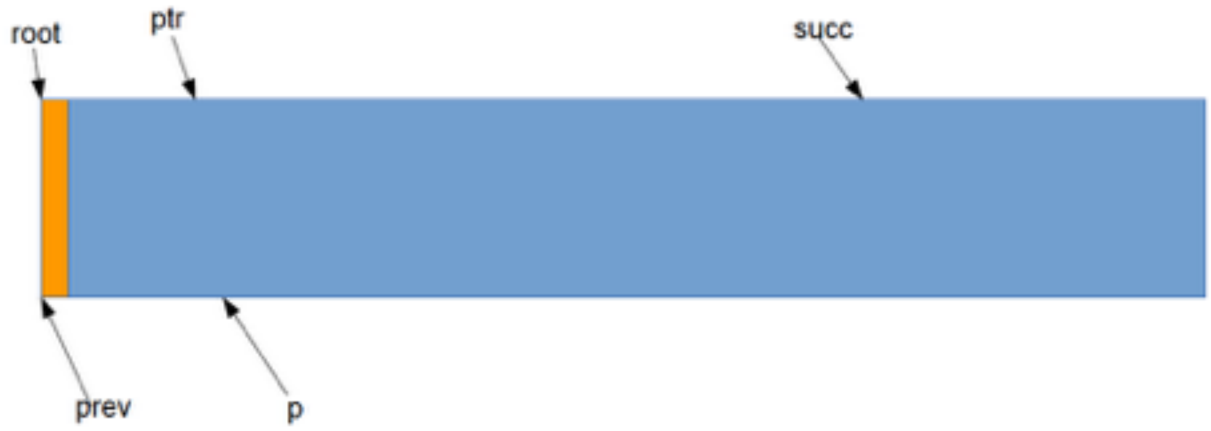
Than next head is created:



Freeing



Merged with the next chunk


root
ptr
succ
prev
p