Assignment 3

Problem: Implement a simulator for memory management for managing free memory in heap, using free list implemented with header block mechanism as discussed in 'Free-Space Management' chapter in OSTEP book. Follow below guidelines for simulator functionality.

1. Your program should provide a command prompt to accept commands one by one like an interpreter, and should be able to process below commands.
    a. *malloc* (example syntax: "**a = malloc(10)**", which allocates 10 units of memory and saves it's pointer in 'a').
    b. *free* (example syntax: "**free(a)**", which frees the memory pointed by 'a').
    c. ***display_status*** (displays the status of the memory (allocated and free, both) and contents of head pointer and other user allocated pointers).
    d. ***coalesce_memory*** (merges adjacent free memory chunks, so if any of the free memory chunk sits right next to one or two other free chunks, merge them into a single larger free chunk).
2. Assume that once memory is handed out to a client (with *malloc* command), it cannot be relocated to another location in memory.
3. Your *free* command should not accept the size of memory to be freed, it just accepts the pointer which is pointing to the memory to be freed.
4. You need not coalesce the memory as soon as a chunk of memory is freed, *coalesce_memory* command should trigger the coalescing process.
5. Header blocks which will be used to manage the free memory should also be kept in the same memory, just before the handed-out chunk of memory. **You should not use any other list to maintain the track of free space, the free space should be tracked using only header blocks.** The header block should contain the size of the allocated or free region and a pointer to the next free memory. Hence the size of header block will be 2 units. So, when a user requests N bytes of memory, the library should **not** search for a free chunk of size N; rather, it should search for a free chunk of size (N+2).
6. You can initialize an array of 100 elements which can be considered as virtual heap of 100 units of memory. Above mentioned commands will manipulate the memory available in this virtual heap and header blocks will also reside in the same.

A sample test run is shown on the next page:

Shell$ *a = malloc(10)*
10 units of memory is allocated pointed by a. Changed pointers: a=2, head=12.
Shell$ *b = malloc(10)*
10 units of memory is allocated pointed by b. Changed pointers: b=14, head=24.
Shell$ *free(a)*
Memory pointed by a (10 units) is freed. Changed pointers: head=0.
Shell$ *display_status*
Free memory track: head=0, 0(10 units)->24(74 units)->null.
Total free memory = 84 units.
User pointers: b=14(10 units).
Shell$ *coalesce_memory*
Memory not coalesced, no adjacent free memory chunks found.
Shell$ *free(b)*
Memory pointed by b (10 units) is freed. Changed pointers: head=12.
Shell$ *display_status*
Free memory track: head=12, 12(10 units)->0(10 units)->24(74 units)->null.
Total free memory = 94 units.
No user pointers at the moment.
Shell$ *coalesce_memory*
Memory coalesced successfully, 4 units of memory saved.
Free memory track: head=0, 0(98 units)->null.
Total free memory: 98 units.