Student Name: Lester Hernandez Alfonso
Panther ID: 4017986
Course: COP4610
Assignment#4

Report

To implement my memory allocator, I defined a block structure consisting of a pointer to the next in line block structure, a block size, and a flag to identify if the block was free, or allocated. The block structure defined, effectively provides the means to define a linked list of block pointers. Since blocks are usually serial in order, and contiguous, then the block structures would be stored at the beginning of a memory block/object, followed by an allocated/unallocated space equivalent to the block size. To traverse to a contiguous next block, one could just use the link found on the block structure of the current block. This way, only a link to the starting block is needed to traverse the whole available memory blocks. Traversing backwards is not necessary, therefore, I did not implement a doubly linked list, but it could have been an option that would have provided some benefits when implementing the merging of contiguous blocks after freeing a block in Mem_Free().

The most difficult part of the assignment, in my opinion was dealing with pointer arithmetic. Since pointers are incremented in units of the type that they point to, every operation with pointers needed to be specified in terms of the size of a block structure. This led to having to allocate block spaces as a multiple of the size of a block structure, otherwise, later operations creating new blocks, or modifying existing ones would lead to inconsistencies; especially when merging free contiguous blocks, or allocating into an existing free block by using some of its memory, while creating a second contiguous block of the size of the leftover memory. In addition to pointer arithmetic, I learned how a basic memory allocator functions, and as a secondary concept I expanded my knowledge of types and typedef in C, including its uses and benefits.

In the following page I will show some output samples obtained from my test program: ‘myprogram’ compiled from ‘testmem.c’, using each of the 3 policies. ‘testmem.c’ can be found in the code directory of this submission.

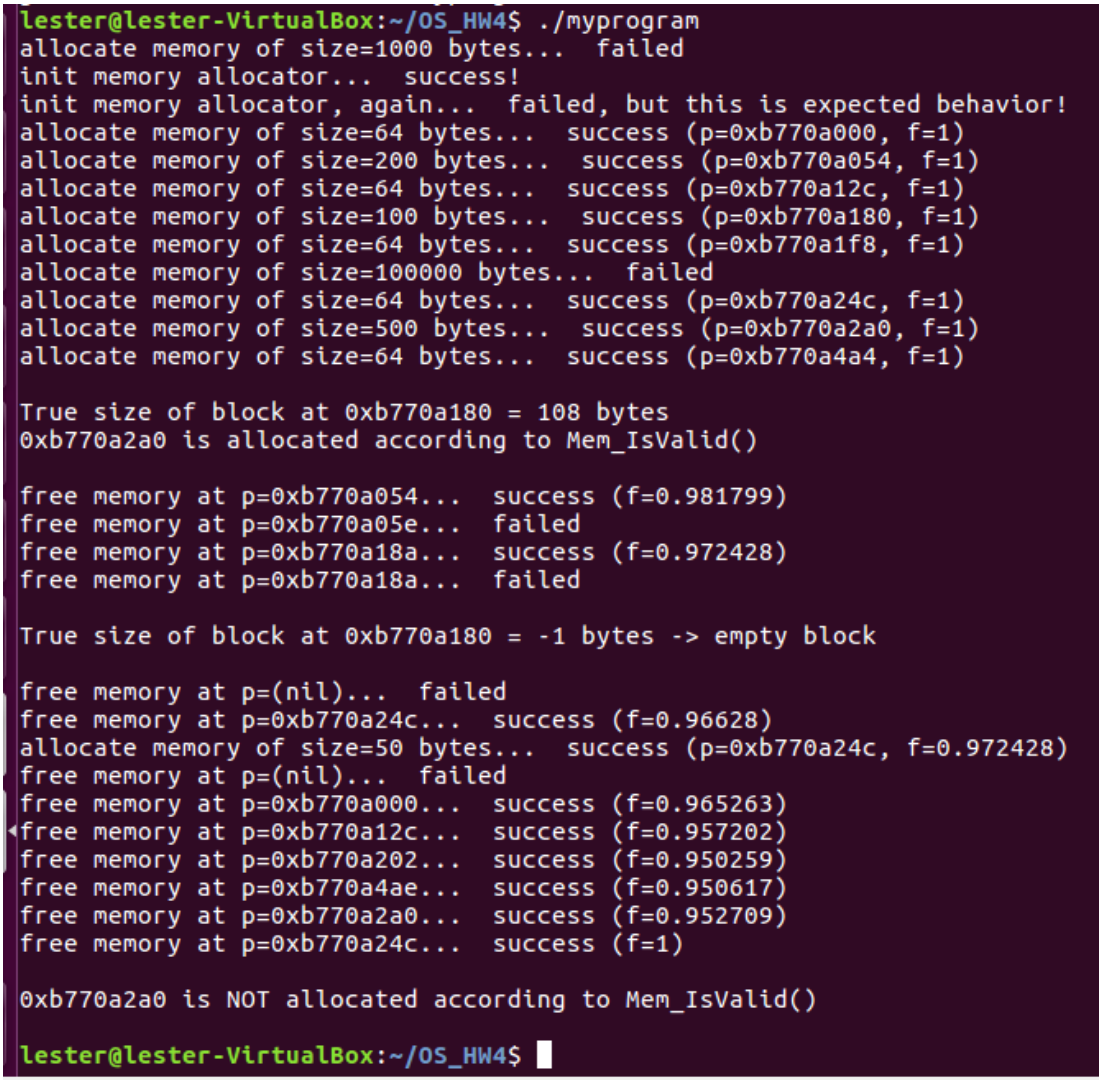
Output Samples:

First-Fit Policy:

```
lester@lester-VirtualBox:~/OS_HW4$ ./myprogram
allocate memory of size=1000 bytes...  failed
init memory allocator...  success!
init memory allocator, again...  failed, but this is expected behavior!
allocate memory of size=64 bytes...  success (p=0xb77c4000, f=1)
allocate memory of size=200 bytes...  success (p=0xb77c4054, f=1)
allocate memory of size=64 bytes...  success (p=0xb77c412c, f=1)
allocate memory of size=100 bytes...  success (p=0xb77c4180, f=1)
allocate memory of size=64 bytes...  success (p=0xb77c41f8, f=1)
allocate memory of size=100000 bytes...  failed
allocate memory of size=64 bytes...  success (p=0xb77c424c, f=1)
allocate memory of size=500 bytes...  success (p=0xb77c42a0, f=1)
allocate memory of size=64 bytes...  success (p=0xb77c44a4, f=1)

True size of block at 0xb77c4180 = 108 bytes
0xb77c42a0 is allocated according to Mem_IsValid()

free memory at p=0xb77c4054...  success (f=0.981799)
free memory at p=0xb77c405e...  failed
free memory at p=0xb77c418a...  success (f=0.972428)
free memory at p=0xb77c418a...  failed

True size of block at 0xb77c4180 = -1 bytes -> empty block

free memory at p=(nil)...  failed
free memory at p=0xb77c424c...  success (f=0.96628)
allocate memory of size=50 bytes...  success (p=0xb77c4054, f=0.972428)
free memory at p=(nil)...  failed
free memory at p=0xb77c4000...  success (f=0.96628)
free memory at p=0xb77c412c...  success (f=0.958203)
free memory at p=0xb77c4202...  success (f=0.950259)
free memory at p=0xb77c44ae...  success (f=0.950617)
free memory at p=0xb77c42a0...  success (f=0.994094)
free memory at p=0xb77c4054...  success (f=1)

0xb77c42a0 is NOT allocated according to Mem_IsValid()

lester@lester-VirtualBox:~/OS_HW4$
```

In line ‘allocate memory of size=50 bytes…’, memory is allocated in the block beginning at address ‘0xb77c4054’. This block is the first free/unallocated block from the start block. However, it is not the best/smallest block for allocation. This behavior is exactly that of the first-fit policy.

Best-Fit Policy:



In this case, the memory of size 50 in the previously mentioned line is allocated in the smallest/best block that it can possibly be in: the block starting at address 0xb770a24c, which has recently been freed.

Worst-Fit Policy:

```
lester@lester-VirtualBox:~/OS_HW4$ ./myprogram
allocate memory of size=1000 bytes...  failed
init memory allocator...  success!
init memory allocator, again...  failed, but this is expected behavior!
allocate memory of size=64 bytes...  success (p=0xb7760000, f=1)
allocate memory of size=200 bytes...  success (p=0xb7760054, f=1)
allocate memory of size=64 bytes...  success (p=0xb776012c, f=1)
allocate memory of size=100 bytes...  success (p=0xb7760180, f=1)
allocate memory of size=64 bytes...  success (p=0xb77601f8, f=1)
allocate memory of size=100000 bytes...  failed
allocate memory of size=64 bytes...  success (p=0xb776024c, f=1)
allocate memory of size=500 bytes...  success (p=0xb77602a0, f=1)
allocate memory of size=64 bytes...  success (p=0xb77604a4, f=1)

True size of block at 0xb7760180 = 108 bytes
0xb77602a0 is allocated according to Mem_IsValid()

free memory at p=0xb7760054...  success (f=0.981799)
free memory at p=0xb776005e...  failed
free memory at p=0xb776018a...  success (f=0.972428)
free memory at p=0xb776018a...  failed

True size of block at 0xb7760180 = -1 bytes -> empty block

free memory at p=(nil)...  failed
free memory at p=0xb776024c...  success (f=0.96628)
allocate memory of size=50 bytes...  success (p=0xb77604f8, f=0.966066)
free memory at p=(nil)...  failed
free memory at p=0xb7760000...  success (f=0.958947)
free memory at p=0xb776012c...  success (f=0.950939)
free memory at p=0xb7760202...  success (f=0.943064)
free memory at p=0xb77604ae...  success (f=0.937243)
free memory at p=0xb77602a0...  success (f=0.896654)
free memory at p=0xb77604f8...  success (f=1)

0xb77602a0 is NOT allocated according to Mem_IsValid()

lester@lester-VirtualBox:~/OS_HW4$
```

In this case, the 50 bytes memory is allocated in the worst/largest possible block; that is, the last block starting at address 0xb77604f8.