Some Practice Heap Problems. Name:

1. Suppose that we receive a series of `malloc( )` requests. The memory allocator enforces 8-byte alignment (*i.e.*, that the allocator will make sure that all blocks have size that's a multiple of 8), and that headers and footers are 4 bytes each. Fill in the remainder of the table:

request	data bytes allocated	block size	block header (in hex)
malloc(9)			
malloc(48)			

2. The table on the back page shows the addresses and contents of some selection of blocks on a heap on a big-endian machine. The header/footer struct is exactly as the one we've described in class, *i.e.*,

```
struct header {
   unsigned int length :29,
   unsigned int NOT_USED :2,
   unsigned int allocated :1 /* 1 means ALLOCATED */
};                           /* 0 means FREE */
```

any additional pointers, if they exist, would be stored in the order: `PREVIOUS`, `NEXT`.

(a) [2 points] What is the address of the header of the first allocated block?

(a) _____

(b) [2 points] What is its length?

(b) _____

(c) [2 points] How much user-data can be stored in this block?

(c) _____

(d) [2 points] What was the address returned by `malloc( )` when this header was set?

(d) _____

(e) [2 points] What is the address of the header of the first free block?

(e) _____

(f) [2 points] What is its length (including header and footers)?

(f) _____

(g) [2 points] How much data could potentially be stored in this block?

(g) _____

(h) [2 points] If this heap uses a simple *explicit free list*, what is the address of the next *free* block?

(h) _____

Address	Value
0x1770	0x00000019
0x1774	0x0000004d
0x1778	0x0000005e
0x177c	0x0000003c
0x1780	0x000000f3
0x1784	0x00000019
0x1788	0x00000018
0x178c	0x00000000
0x1790	0x000017c0
0x1794	0x00000073
0x1798	0x00000005
0x179c	0x00000018
0x17a0	0x00000021
0x17a4	0x000000f7
0x17a8	0x0000008c
0x17ac	0x000000f7
0x17b0	0x0000003e
0x17b4	0x000000c9
0x17b8	0x00000074
0x17bc	0x00000021
0x17c0	0x00000010
0x17c4	0x00001788
0x17c8	0x00000000
0x17cc	0x00000010