Dynamic Memory Allocation Name: _____

Important Information

- The memory allocator enforces **8-byte alignment**.
  - i.e. The allocator will make sure that **all blocks have a size that is a multiple of 8 bytes**.
- **Headers** and **footers** are **each 4 bytes**.
  - So, the **total bytes allocated** will equal the **number of bytes to be allocated for data plus the additional 8 bytes** (4 bytes each) necessary to store the header and the footer.
- The **header struct** is defined below (uses bit-fields, see K&R section 6.9 for more information).
  - The footer consists of the same information as the header, so **the same struct will be used for both the header and the footer**.

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

- To keep track of free blocks, we will use a **simple explicit free list**.
  - So, following the header of a free block, the **pointers** to **the previous free block** and to **the next free block** will be found in that order.
- Assume that we are looking at a **big-endian** machine.

Problems

1. Using the relevant information given above, fill in the remainder of the table below.

Request	Bytes Allocated for Data	Total Bytes Allocated (block size)	Value of Header (in hex)
malloc(9)			
malloc(48)			

Dynamic Memory Allocation Name: _____

2. The table below shows the addresses and contents of some selection of blocks on a heap (remember we are assuming a big-endian machine). Use this table to answer the following questions (a-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

a. What is the address of the header of the first allocated block?

b. What is its length?

c. How much user-data can be stored in this block?

d. What was the address returned by malloc( ) when this header was set?

e. What is the address of the header of the first free block?

f. What is its length (including header and footers)?

g. How much data could potentially be stored in this block?

h. Given that this heap uses a simple explicit free list, what is the address of the next free block?