Pointers are four bytes and pages are 4K. Here's the heap. **Malloc()** and **free()** have been implemented as described in the last class. The head of the free list is `0x58a00`. (And if you don't remember, the definition of the free list structs are to the right).

```
struct flist {
  int size;
  struct flist *flink;
  struct flist *blink;
};
```

You'll find the printout that I gave you to be helpful.

```
Address  Value      Address  Value      Address  Value      Address  Value
-------  -------    -------  -------    -------  -------    -------  -------
0x589a0  0x00010    0x589d0  0x00018    0x58a00  0x00030    0x58a30  0x00010
0x589a4  0x589d0    0x589d4  0x00000    0x58a04  0x589a0    0x58a34  0x589a0
0x589a8  0x58a00    0x589d8  0x589a0    0x58a08  0x00000    0x58a38  0x589c8
0x589ac  0x00018    0x589dc  0x589e0    0x58a0c  0x58a1c    0x58a3c  0x00020
0x589b0  0x00020    0x589e0  0x58a3c    0x58a10  0x58a24    0x58a40  0x00018
0x589b4  0x00018    0x589e4  0x00018    0x58a14  0x58a1c    0x58a44  0x589c0
0x589b8  0x589e0    0x589e8  0x00018    0x58a18  0x58a00    0x58a48  0x58a18
0x589bc  0x58a08    0x589ec  0x589c0    0x58a1c  0x589cc    0x58a4c  0x00010
0x589c0  0x58a44    0x589f0  0x58a1c    0x58a20  0x58a20    0x58a50  0x00008
0x589c4  0x589cc    0x589f4  0x00020    0x58a24  0x00030    0x58a54  0x589a0
0x589c8  0x00010    0x589f8  0x58a30    0x58a28  0x00018    0x58a58  0x589a0
0x589cc  0x589ec    0x589fc  0x58a08    0x58a2c  0x58a3c    0x58a5c  0x58a24
```

Please answer the following questions:
Question 1: How many bytes are in the first node (or memory chunk) on the free list?
Question 2: What is the memory address of the second node on the free list?
Question 3: How many bytes are in the second node (or memory chunk) on the free list?
Question 4: How many nodes are there on the free list?
Question 5: How many allocated chunks are there (memory chunks that have been malloc'd, but not freed)?
Question 6: If I call **sbrk(0)**, what will it return (It is not 0x58a60)?

Answer to the clicker questions

To explain this question, first, identify the nodes on the free list:

- Node 1: Starts at 0x58a00 and is 0x30 = 48 bytes in size. Why 0x30? Because the first four bytes are the size. The next four bytes are **flink**, so the next node on the free list is 0x589a0.

- Node 2: Starts at 0x589a0 and is 0x10 = 16 bytes in size. Its **flink** value is 0x589d0, so that's the next node on the free list.

- Node 3: Starts at 0x589d0 and is 0x18 = 24 bytes in size. Its **flink** value is 0x00000, so that's the end of the free list.

That gives you the answer to questions 1 - 4:

- **Question 1**: 0x30 or 48.
- **Question 2**: 0x589a0.
- **Question 3**: 0x10 or 16.
- **Question 4**: 3.

Now, to answer the rest of the question, it's helpful to differentiate the free memory from the allocated memory. I've done that below by coloring the free memory blue:

Address	Value
0x589a0	0x00010
0x589a4	0x589d0
0x589a8	0x58a00
0x589ac	0x00018
0x589b0	0x00020
0x589b4	0x00018
0x589b8	0x589e0
0x589bc	0x58a08
0x589c0	0x58a44
0x589c4	0x589cc
0x589c8	0x00010
0x589cc	0x589ec
0x589d0	0x00018
0x589d4	0x00000
0x589d8	0x589a0
0x589dc	0x589e0
0x589e0	0x58a3c
0x589e4	0x00018
0x589e8	0x00018
0x589ec	0x589c0
0x589f0	0x58a1c
0x589f4	0x00020
0x589f8	0x58a30
0x589fc	0x58a08
0x58a00	0x00030
0x58a04	0x589a0
0x58a08	0x00000
0x58a0c	0x58a1c
0x58a10	0x58a24
0x58a14	0x58a1c
0x58a18	0x58a00
0x58a1c	0x589cc
0x58a20	0x58a20
0x58a24	0x00030
0x58a28	0x00018
0x58a2c	0x58a3c
0x58a30	0x00010
0x58a34	0x589a0
0x58a38	0x589c8
0x58a3c	0x00020
0x58a40	0x00018
0x58a44	0x589c0
0x58a48	0x58a18
0x58a4c	0x00010
0x58a50	0x00008
0x58a54	0x589a0
0x58a58	0x589a0
0x58a5c	0x58a24

The allocated memory will be the memory in the black font. The first 4 bytes of a chunk will be the chunk's size. So:

- Chunk 1: Address = 0x589b0 and Size = 0x20 or 32 bytes. Therefore, the chunk goes right up to the free chunk at 0x589d0.
- Chunk 2: Address = 0x589e8 and Size = 0x18 or 24 bytes. Therefore, the chunk goes right up to the free chunk at 0x58a00.
- Chunk 3: Address = 0x58a30 and Size = 0x10 or 16 bytes. This means that the next chunk starts at 0x58a40.
- Chunk 4: Address = 0x58a40 and Size = 0x18 or 24 bytes. This means that the next chunk starts at 0x58a58. Or not -- because there's not a valid size there, and the heap "ends" after 0x58a5c. So this is the last chunk, and what we're printing are eight bytes that are on the same page, but have not been assigned to the process by the operating system yet. This clues us into the answer to the last question.

- **Question 5**: 4
- **Question 6**: It will return 0x58a58. If our page sizes are 4K, then the addresses up to 0x58fff will not seg fault, but they are not officially allocated to our process yet.