Behold the procedure **a()**:

```
typedef unsigned long UL;

void a(unsigned long *d,
       unsigned int  *j,
       unsigned int  *k)
{
  int i;
  printf("d    = 0x%016lx\n", (UL) d);
  printf("j    = 0x%016lx\n", (UL) j);
  printf("k    = 0x%016lx\n\n", (UL) k);
  printf("*j   = 0x%08x\n", *j);
  printf("*k   = 0x%08x\n\n", *k);

  for (i = 0; i < 10; i++) {
    printf("d[%d] = 0x%016lx\n", i, d[i]);
  }
}
```

When I run this, I get the following output:

```
d    = 0x00007f9cec401798
j    = 0x00007f9cec4017e8
k    = 0x00007f9cec401790

*j   = 0xe3c017ff
*k   = 0x00c01705

d[0] = 0x00007f9cec4017c8
d[1] = 0x00007f9cec4017e8
d[2] = 0x00007f9cec4017e0
d[3] = 0x00007f9cec401798
d[4] = 0x00007f9cec4017d8
d[5] = 0x00007f9cec4017b0
d[6] = 0x00007f9cec4017c8
d[7] = 0x00007f9cec4017e8
d[8] = 0x00007f9cec401798
d[9] = 0x00007f9cec4017d8
```

My machine has 8 byte pointers and is in little endian.
Please answer the following questions:

Q1: What is the byte, in hex, at address `0x00007f9cec4017e8`?

Q2: What is the byte, in hex, at address `0x00007f9cec4017e9`?

Q3: What is the byte, in hex, at address `0x00007f9cec4017a8`?

Q4: What will the following **printf()** statement print?

```
printf("0x%08lx\n", k[4]);
```

Q5: What will the following **printf()** statement print?

```
unsigned int **x;

x = (unsigned int **) d[5]
printf("0x%08x\n", **x);
```

Answers Without Explanation

```
Q1: 0xff
Q2: 0x17
Q3: 0xe0
Q4: 0xec4017e8
Q5: 0xec4017c8
```

Answering without drawing out memory

My first set of answers will just use logic and knowledge of pointers. It's how I would answer the question if I were given this question as a clicker question. The second set of answers requires more work -- I'll draw out everything I know about memory. It's how I would answer the question on an exam, where I have more time and *really* want to make sure I get it right.

Question 1

This address is equal to **j**, so this is the first byte of ***j**. Remember that the machine is little endian, so the first byte of ***j** is 0xff.

Question 2

This is the second byte of ***j**; 0x17.

Question 3

We need to use **d** to answer this. The pointer value in the question is 16 bytes greater than **d**. So, this is the first byte of **d[2]**. Again -- little endian -- so 0xe0.

Question 4

k[4] is equal to ***(k+4)**. Since each element of **k** is four bytes, **(k+4)** is 0x00007f9cec4017a0. That's eight bytes more than **d**, so this is the first four bytes of **d[1]**: 0xec4017e8.

Question 5

The hardest question. **d[5]** is 0x00007f9cec4017b0, which is **(d+3)**. So, ***x** is equal to **d[3]**, which is 0x00007f9cec401798. That address is equal to **d**, so ****x** is equal to the first four bytes of **d[0]**: 0xec4017c8.

Answering by drawing out memory

On an exam, I would cut-and-paste **d**, and then add everything I know about **d**, **j** and **k** (and even **x**). I'll put *'s for bytes I don't know. I'll also label the bytes with their byte number in hex, so that the little endian is less confusing. From this labeled drawing, I can get all of the answers. It may help you understand them, too, so this is a good exercise for you:

```
   What           Addresss           What                Value
-------------- ------------------ -----------------    ------------------
                                                          7 6 5 4 3 2 0 1
   (k+0)       0x00007f9cec401790     k[1] k[0]       = 0x********00c01705
*x (k+2) (d+0) 0x00007f9cec401798 **x k[3] k[2] d[0] = 0x00007f9cec4017c8
   (k+4) (d+1) 0x00007f9cec4017a0     k[5] k[4] d[1] = 0x00007f9cec4017e8
         (d+2) 0x00007f9cec4017a8               d[2] = 0x00007f9cec4017e0
       x (d+3) 0x00007f9cec4017b0            *x d[3] = 0x00007f9cec401798
         (d+4) 0x00007f9cec4017b8               d[4] = 0x00007f9cec4017d8
         (d+5) 0x00007f9cec4017c0             x d[5] = 0x00007f9cec4017b0
         (d+6) 0x00007f9cec4017c8               d[6] = 0x00007f9cec4017c8
         (d+7) 0x00007f9cec4017d0               d[7] = 0x00007f9cec4017e8
         (d+8) 0x00007f9cec4017d8               d[8] = 0x00007f9cec401798
         (d+9) 0x00007f9cec4017e0               d[9] = 0x00007f9cec4017d8
             j 0x00007f9cec4017e8                *j  = 0x********e3c017ff
```

Question 1
Question 2
Question 3
Question 4
Question 5