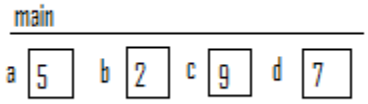
Pass by Reference Tracing Example

In debugging and creating code, it's imperative to understand the process the computer uses with both pass by value and pass by reference parameters. The following example illustrates the details of many situations that might arise. Keep in mind that the following code is NOT meant to be written to solve a problem. In fact, it's intentionally written to be confusing. But, if one can trace through this example, she should be able to understand what is transpiring in any situation with function calls, at least with respect to the mechanics of what is going on.

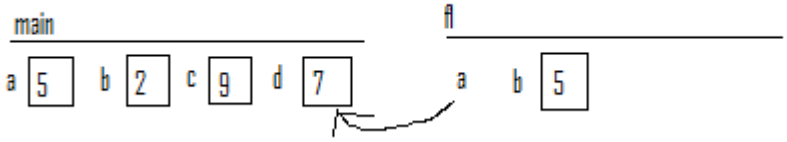
Here is the program:

```
#include <stdio.h>

int f1(int *a, int b);
int f2(int a, int *b);

int main() {

    int a = 5, b = 2, c = 7, d = 9;                 // line 8

    c = f1(&d, a);                                  // line 10
    printf("a=%d,b=%d,c=%d,d=%d\n",a,b,c,d);        // line 11
    a = f2(c - d, &a);                              // line 12
    printf("a=%d,b=%d,c=%d,d=%d\n",a,b,c,d);        // line 13
    b = f1(&c, 8);                                  // line 14
    printf("a=%d,b=%d,c=%d,d=%d\n",a,b,c,d);        // line 15
    d = f2(b, &a);                                  // line 16
    printf("a=%d,b=%d,c=%d,d=%d\n",a,b,c,d);        // line 17
    return 0;
}

int f1(int *a, int b) {                             // line 21
    *a = b - 8;                                     // line 22
    b = b*2 - (*a);                                 // line 23
    printf("a=%d,b=%d\n",*a,b);                     // line 24
    return b - *a;                                  // line 25
}

int f2(int a, int *b) {                             // line 28
    a = *b + a;                                     // line 29
    *b = 37 - *b;                                   // line 30
    printf("a=%d,b=%d\n",a,*b);                     // line 31
    return a;                                       // line 32
}
```

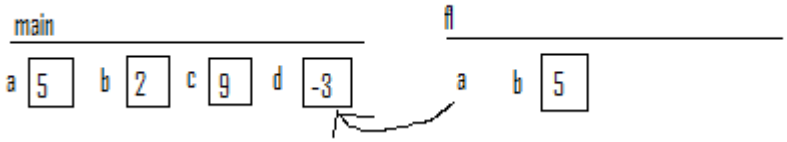
After line 8 is executed, our picture in memory is as follows:



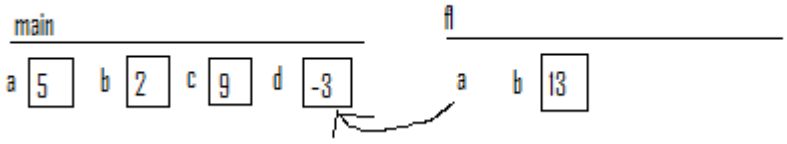
As we make the function call on line 10, here is the picture right after the actual parameter values (memory address of d in main, and 5) from main have been copied into the formal parameters for f1:



Now, we run line 22 in f1. This line calculates b – 8, which is -3, and stores that into *a, the box that stores the variable d in main, since the pointer a, in f, points to the memory address where the variable d, in main, is stored. Thus, after line 22, our picture is:



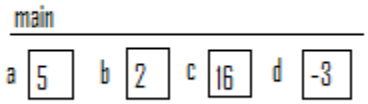
Line 23 calculates 2*b – (*a), plugging in 5 for b and -3 for *a, obtaining 13, which then gets stored into b:



At this point the following gets printed out:

```
a=-3,b=13
```

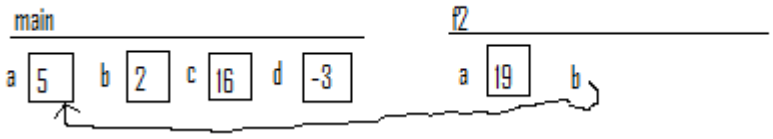
The value that gets returned to main is 13 – (–3) = 16. As this value is being returned, the memory for the function disappears. As 16 arrives to main, main uses the assignment statement in line 10 to store 16 into the variable c, in main:



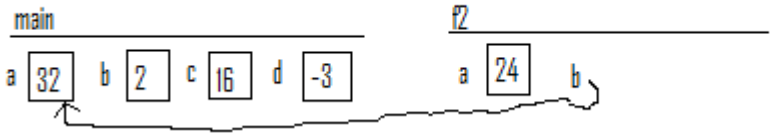
Thus, the second line of output produced by the program is:

```
a=5,b=2,c=16,d=-3
```

As we make the function call to f2 on line 12, our picture looks like the following after copying over the actual parameters (19 and the address of the variable a in main), into the corresponding formal parameters:



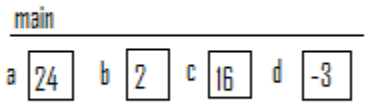
Now, we run line 29, which changes the variable a to 5+19 = 24, since the value of the variable that b from f2 is pointing to is 5. When we run line 30, it changes the variable b from f2 is pointing to to 37 – 5 = 32. Thus, after both those lines of code, our picture is as follows:



Thus, the third line of output produced by the program is:

```
a=24,b=32
```

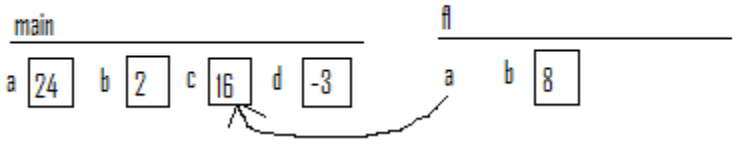
The value returned to main is 24, and this value gets stored in the variable a, in main, by the assignment statement in line 12:



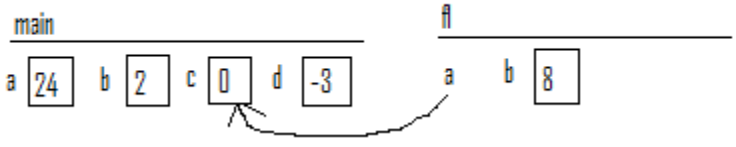
Thus, the fourth line of output produced by the program is:

```
a=24,b=2,c=16,d=-3
```

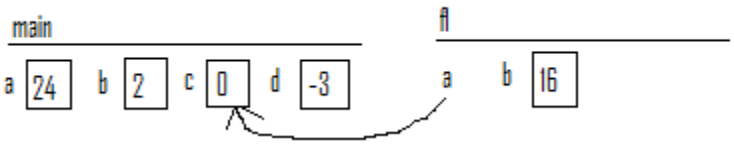
In line 14, after we copy over the values of the actual parameters from main into the formal parameters of f1, we get the following:



Line 22 changes the value of the variable to which the pointer a points to 0:



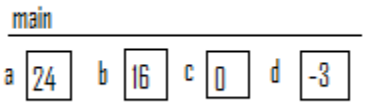
Line 23 changes b in f1 to 16, since 2*8 – 0 = 16:



Then the function prints:

```
a=0,b=16
```

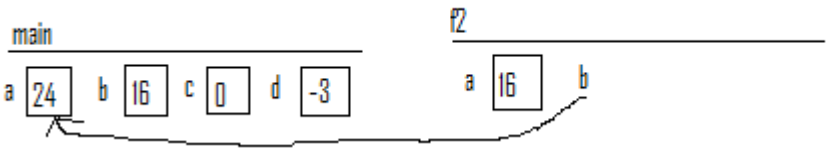
Then the value 16 gets returned to main, where it is stored in the variable, b, in main:



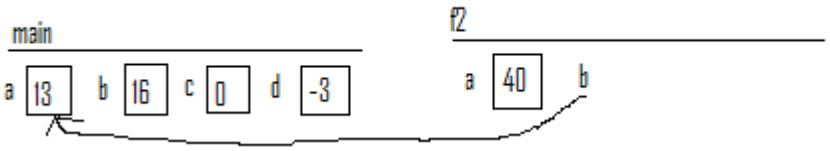
Main prints:

```
a=24,b=16,c=0,d=-3
```

As the we start the last function call on line 16, after the actual parameter values of 16 and the address of the variable a are copied into the formal parameters in f2, we have:



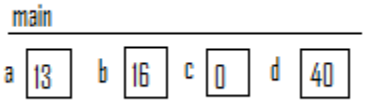
Line 29 will change the variable a in f2 to 40, since 24+16 = 40. Line 30 will change the variable to which a points to 37 – 24 = 13. At this point we have:



Thus, the seventh line printed by this program is:

```
a=40,b=13
```

Line 32 returns the value 40 to main, which gets stored in the variable d in main:



The final line of output is:

```
a=13,b=16,c=0,d=40
```

Recapping, the entire output of this program is:

```
a=-3,b=13
a=5,b=2,c=16,d=-3
a=24,b=32
a=24,b=2,c=16,d=-3
a=0,b=16
a=24,b=16,c=0,d=-3
a=40,b=13
a=13,b=16,c=0,d=40
```