Make a fair coin from a biased coin

You are given a function foo() that represents a biased coin. When foo() is called, it returns 0 with 60% probability, and 1 with 40% probability. Write a new function that returns 0 and 1 with 50% probability each. Your function should use only foo(), no other library method.

Solution:
We know foo() returns 0 with 60% probability. How can we ensure that 0 and 1 are returned with 50% probability?
The solution is similar to this post. If we can somehow get two cases with equal probability, then we are done. We call foo() two times. Both calls will return 0 with 60% probability. So the two pairs (0, 1) and (1, 0) will be generated with equal probability from two calls of foo(). Let us see how.

(0, 1): The probability to get 0 followed by 1 from two calls of foo() = 0.6 * 0.4 = 0.24
(1, 0): The probability to get 1 followed by 0 from two calls of foo() = 0.4 * 0.6 = 0.24

So the two cases appear with equal probability. The idea is to return consider only the above two cases, return 0 in one case, return 1 in other case. For other cases [(0, 0) and (1, 1)], recur until you end up in any of the above two cases.

The below program depicts how we can use foo() to return 0 and 1 with equal probability.

```
#include <stdio.h>

int foo() // given method that returns 0 with 60% probab
{
    // some code here
}

// returns both 0 and 1 with 50% probability
int my_fun()
{
    int val1 = foo();
    int val2 = foo();
    if (val1 == 0 && val2 == 1)
        return 0;   // Will reach here with 0.24 probabi
    if (val1 == 1 && val2 == 0)
        return 1;   // // Will reach here with 0.24 prob
    return my_fun();  // will reach here with (1 - 0.24
}

int main()
{
    printf ("%d ", my_fun());
    return 0;
}
```

References:

http://en.wikipedia.org/wiki/Fair_coin#Fair_results_from_a_biased_coin