1. Since the first part of the statement is false the entire and statement will be marked as false regardless the result of the casting of the null pointer. So the program will skip it and just say the statement is false, so someFunc() returns 0.
2. Well at low numbers the effects of the recursion are not really noticeable. But, consider what happens when we reach the 100th or 1000th term. Just to print this one term the function will call itself twice every time the result is greater than 1. This means that the if were to get the 1000th term the function would call itself and twice and each of those functions calls two more functions and the next set calls another two each to the point where the bottom level has 2^n number of functions called and processed before the function can even process the functions above those. All of these function call add their current values are stored in memory. In addition the calculation for every single number requires massive amounts of processing time. The better solution is to just know calculation each solution beforehand individually and pull n-1 and n-2 out of memory verses trying to process all of the results every time a request is made.
3. For this problem in addition to researching how branch prediction works, I found finding a C to assembly converter can help you see how the program branches. So because the way branch prediction uses previous branches as an example for what the most likely result will be, this code is very helpful to a branch prediction mechanism. This code uses for loops when it branches which has a single address that gets used repeatedly and an exit one that gets us once. Also due to the clearly defined iterators many modern processors also used that to help predict when that branch will exit. The only hurtful part of this example is that it uses two for loops, which in some processors could be too much to store in the very small Branch Target Buffer (BTB). That’s dependent of the CPU though from my understanding. The Ideal case for this program in terms of effectively using the BTB would be to use only one for loop for the iteration process.