Rob Navarro
December 1, 2015

Lab 10

For this lab I used the clock function to measure the amount of clicks that occurred when each function ran. I found that when my program compiled on the flip server the functions were running so fast that I was unable to measure a time. Because of this I ended up running the double recursion function an extra 100000 times and the iteration function an extra 1000000 times. This allowed me to have measurable times and once I had a time I would divide by the total iterations and then I had the actual times.

I have done this type of exercise before so I knew that the recursive function would be slower than the iteration function. What I ended up finding was that for very low numbers the values were almost identical but the recursive function did run slightly slower. As I continued to increase the number the recursive function started to take much longer. For example, when I calculated the 15^{th} Fibonacci number the iteration function took only 4E-5 milliseconds while the recursive function took 0.0134999 milliseconds. That is quite the time difference and proves that the recursive function is far less efficient.

For the tail recursive and non tail recursive I expected the tail recursive function to be faster. When I first started to mess with these two function I decided to leave the optimization off. While testing the tail and non tail functions I found that the functions were running so fast that they were basically immeasurable on the millisecond scale. I think this had to do with the optimization being turned on. Without it on the tail function was just a bit faster than the non tail function.

After reviewing all of these functions I found that for the 12^{th} number, with no optimization turned on, the iteration function was actually still the fastest with a time of 3E-5 milliseconds. It was followed by the non tail function (4E-5 milliseconds), then the tail function (6E-5 milliseconds), and finally the double recursive function (0.0029997 milliseconds). When I ran these functions with optimization turned on they all ran so fast that I was unable to get any measurable results. I suppose this proves that the two tail functions need optimization to run as fast as possible, which is hinted at in the lab description. It also shows that all of the functions will run extremely quickly if optimization is turned on.

Overall I was a bit surprised to see that iteration ran the fastest with optimization turned off. I do remember previously reading that while recursion looks nice, it doesn't necessarily run efficiently. I think this lab does a good job of proving that.