David Alummoottil Readme – Formula

In order to complete formula in pa3, I had three major functions; the main, nCr, and Factorial, the latter two of which I attempted to optimize. The main function, which is written in C, checks for errors in the input and formats the final result that is wanted, 1+nC1*x^1 + nC2*x^2...+ nCr*x^r+...nCn*x^n. First I check for errors like if the input is the correct number of arguments, or if there are invalid characters entered. My program strictly only allows digits 0-9 into the program, and any other characters are considered invalid. However, it does allow the user to input the help flag, which results in a corresponding message for the user. Once it verifies the input, the main calls the nCr function, written in assembly, to calculate the coefficients. nCr in return takes n and r as parameters and calls Factorial to figure out what nCr would be. The formula nCr uses is n!/r!(n-r)! . Factorial uses a for loop to calculate the factorial of the input, and also checks if overflow occurs when calculating the answer, and if overflow does occur, returns 0. nCr then checks if the returned answer is 0 and returns 0 if it is, indicating overflow. It also checks in its own calculations with the results of factorial if an overflow occurred and returns 0 if it did. Back in the main, it takes the returned value of nCr and inserts it into the print statement that is outputted to the user as the answer. I checked the runtime of the program by using the system call gettimeofday() starting from once the input was verified as proper and ending when the program was done. Both factorial and nCr were written in assembly x86-AT&T 32-bit format. As a result, my program will not be able to work with large numbers that a 64-bit machine could handle. One challenge I faced was figuring out how to implement a check for overflow in the assembly code. I also needed to figure out how to properly call Factorial within nCr. This program ran in O(n)

time because of the way it was designed and implemented. Since there are n coefficients that must be calculated, the program will run n times. Memory was not dynamically allocated in this program either.