1. [6 points] Recursion. Read the assigned chapter and notes for Week 1 located in the Learning Activities area in Blackboard. Then provide solutions for the following:

(a). [3 points] Download the g.cpp file, then using the definition below, implement the details of a recursive function called g(n).

*Attached g.cpp*

(b). [1 point] What type of recursion does the g() function in part (a) use? Briefly explain your answer.

*The function g() in g.cpp uses a type of recursion called tail recursion. This is situation where the subroutine is called as the final action of a procedure. In part (a), the value of n determines whether a subroutine is called as a tail recursion or if 0 is returned.*

(c). [2 points] Perform an Internet search and provide a brief description (at least one paragraph with four to five sentences) of an example of a practical computer application use for recursion. For example, the practical use you mention should be something other than simple function implementations such as factorial, the power of a number, Fibonacci, etc. The description must be summary in your own words and not a cut and paste from a website. Include the reference to your source or sources in APA format at the end of your description for this problem.

*In the Linux/Unix operating system, a program called grep is used to search for either a filename, or contents of a file. The power of grep can be fully implemented when using the “-r” option to search recursively. When option “-r” is applied, grep searches this current directory, then proceeds to the next available subdirectory in a depth-first search fashion until no other subdirectories remain. After the recursive search is performed, the program can then list the locations of the searched item in the terminal.*

**GNU Grep 2.25. (n.d.). Retrieved May 30, 2016, from** [**https://www.gnu.org/savannah-checkouts/gnu/grep/manual/grep.html**](https://www.gnu.org/savannah-checkouts/gnu/grep/manual/grep.html)

**2.** [4 points] Complexity Analysis. Begin by reading the assigned chapter and notes for Week 2 located in the Learning Activities area in Blackboard. Then answer the following questions:

(a) [2 points] What is the asymptotic complexity (or big-O) of the section of code used for adding two 𝑛 𝑥 𝑛 matrices b and c, with the result going into matrix a?

for (int i=0; i < n; i++){

for (int j=0; j < n; j++){

a[i][j] = b[i][j] + c[i][j];

}

}

*The complexity of this section of code is O(n^2) since the program would compute the addition of each row of “n” numbers for all “n” columns. With 5 rows and 5 columns, there would be 25 additions performed.*

(b) ) [2 points] Briefly explain why is it not necessarily desirable to use real-time measures (e.g. nanoseconds, seconds, etc.) to determine the efficiency of an algorithm.

*It is insufficient to use real-time measurements for determining the efficiency of an algorithm because of varying degrees of computation power from computer to computer. A algorithm will run faster on a more powerful computer than on a less powerful computer.*