Problem Set 1: Recursion and Complexity Analysis

CS3330 Data Structures and Algorithms
Term 1 2016: August 15 – October 15
Dr. Jack Davault

Overview: For problems 1(a) and 2(b) of this assignment, you will need a C++ compiler. In order to receive credit, your programs must compile and run; and you must provide the actual source code file so that I can compile and run your program (e.g. the file you modified or created ending in **.cpp**). Examples on how to import existing source code files into your compiler are provided in the file called **Importing Source Code.pdf**. The remaining problems for the assignment must be written up within a single Microsoft Word document. You must include your name and course number within all files that you submit, including source code files as a comment at the top of each file that you create or modify.

1. **[4 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] A palindrome is a word or phrase that is spelled the same both forward and backward. For example, "civic" is a palindrome. For this problem, implement the details for a recursive function called `validPalindrome(s)`, where `s` is a given string letters and/or spaces. The `validPalindrome()` function is a predicate function that returns either true (1) or false (0), depending on whether or not a given word is a palindrome. The implementation for this function must use recursion and can be directly translated into C/C++ from the following mathematical definition:

$$validPalindrome(s) = \begin{cases} 1 & \text{if } |s| <= 1 \\ validPalindrome(substring(1, |s| - 2) & \text{if } s[0] = s[|s| - 1] \\ 0 & \text{otherwise} \end{cases}$$

   Hint: (1) In mathematics $|s|$ is read "the order of s", which means the number of items in s. Since s is a string this translates to the number of characters in s. In C++ this can be translated into code as `s.length()`. (2) The `substr()` function is a built-in method used to return a substring of a given string. You can call this function on s as you did with the `length()` function. For example: `s.substr()`, but pass in the appropriate parameters.

   Output: The output for program once the function is implemented will print each word and 1 or 0 depending on whether or not the word is a palindrome.

   Solution: See an example implementation of this function in the **palindrome.cpp** file in the **palindrome.zip** file.

   (b) [1 point] Based on the readings and materials in the Learning Activities area, what type of recursion does the `validPalindrome()` function in part (a) use? Briefly explain your answer.

Solution: Because the recursive statement is not that last step in the recursive function, the validPalindrom() function is an example of Nontail Recursion. You also received credit if you said that this function is an example of Direct Recursion.

2. [6 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] Briefly explain the difference between Θ (Theta) notation and big-Ω notation. Also provide the mathematical definitions of each**.**

Solution: Θ-notation combines big-O and big-Ω in order produce what is known as a tight bound of a function. Tight bound means that the function will never exceed a point *N* and will never go below a point *N*. Big-Ω on the other hand measures the lower bound. Lower bound means that a never go below a point *N*. Remember that some may define the fixed point *N* as n_0

The formal definition of Θ-notation is: $f(n)$ is $\Theta(g(n))$ if there are positive numbers c_1, c_2 and N such that $c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge N$.

And the formal definition of big-Ω is: $f(n)$ is $\Omega(g(n))$ if there exists positive numbers *c* and *N* such that $f(n) \ge cg(n)$ for all $n \ge N$.

(b) [1 points] What is the asymptotic complexity (or big-O) of the following block of code?

Hint: No programming is necessary for this problem. Just tell me the complexity of the function, and provide a brief explanation on how you arrived at the solution.

```
int cubes(int n)
{
    for (int i=1; i <= n; i++) {
       cout << i * i * i << " " << endl;
       |
}
```

Solution: This function has a single loop. The for-loop executes *n* times. Therefore the big-O or complexity of this function is *O*(*n*).

(c) [3 points] Write a program that will determine is a given number is prime. Your main program should pass an integer to a function `isPrime(int n)` to determine if the given number is prime. The `isPrime(int n)` function should return true if the passed in value is a prime number or false otherwise. The function `main()` should print the number and word "TRUE" if is a prime or "FALSE" otherwise.

Solution: See an implementation for this problem in the **prime.cpp** file.

Other Notes: I recommend submitted your solutions as a single Zip file using the Problem Set 1 link provided in the Assignments area. If you are using the Visual C++ or Dev-C++ compiler, you should only submit the source code files for your program (the files ending in **.cpp**). For space

reasons, please do not submit the entire Visual C++ or Dev-C++ project folders. Do not hesitate to ask if you have any questions or need clarification on what to do for this assignment.