CSE 122 – HW 0 Rubric

Student Name: Noah Cherry

Grader Name: Kyle Buchmiller

Questions	Score	Total
Code Requirements		
1.) FizzBuzz		

Requirements:
 • Diff with solution fizzbuzz.sol (**4 pts**)
   o This is all or nothing. **-2 you started at 0 instead of 1**
 • Used the algorithm correctly (**10 pts**)
   o Correct code, even if the output doesnt match with diff.
 • Naming is correct (**1 pts**)
   o Source code "**fizzbuzz.c**"
   o Out put file "**fizzbuzz.out**"
	13	15

2.) Function Timing		

Requirements:
 • Solution should be similar to **sum.sol** (**8 pts**)
   o This **can't** be diffed.
   o Everything but the last column should be the same as sum.sol
 • Used the algorithm correctly (**6pts**)
   o Used a **for** loop
   o Used unsigned ints for input and output
   o Time should be printed to 24 decimal places
 • Naming is correct (**1 pt**)
   o Source code " **sum.c** "
   o Output file "**sum.out**"
	15	15

3.) Function Timing – Non Iterative		

Requirements:
 • Solution should be similar to **sum_no_for.sol** (**8 pts**)
   o This **can't** be diffed.
   o Everything but the last column should be the same as sum_no_for.sol
   o **-2 first column in your .out shouldn't exist**
   o
 • Used the algorithm correctly (**6pts**)
   o Uses the equation to calculate the sum (no loops)
   o uses unsigned ints for input and output
	13	15

• Naming is correct (**1 pts**)
    o Source code " **sum_no_for.c** "
    o Output file "**sum_no_for.out**"		

4.) Written Answers – to_for_or_not_to_for.txt		

Requirements:
• Ask this for the questions 2 and 3. (**5 pts each**)
    o What is the largest k before overflow?
    o Is k the same for both functions?
    o Which is preferred? Why? -2 why do you prefer the non-for method? There's an important reason for why the non-for method is better
• Should be reasonable answers (no for loop preferred, overflow with the for loop).
	8	10

5.) Cube Root		

Requirements:
• Diff with solution cube_binary.sol (**4 pts**) **-4 differs**
    o This is all or nothing.
• Used the algorithm correctly (**10 pts**)
    o Finds the cube root of the numbers 1-100
    o Accuracy is to 9 decimal places -1 should be for 9 decimal places
    o Uses **binary search** (6/10 pts)
• Naming is correct (**1 pts**)
    o Source code "**cube_binary.c**"
    o Output file "**cube_binary.out**"
	10	15

6.) Newton's Square Root		

Requirements:
• Diff with solution newton_sqrt.sol (**3 pts**) **-3 differs**
    o This is all or nothing.
• Used the algorithm correctly (**6 pts**)
    o Uses Newton's Method (5/6 pts)
    o Accuracy is to 9 decimal places -1 yours is 12 decimal places
• Naming is correct (**1 pts**)
    o Source code " newton_sqrt **.c**"
    o Output file " newton_sqrt.**out**"
	11	10

CSE 122 – HW 0 Rubric

7.) Newton's Cube Root		

Requirements:
• Diff with solution newton_cube.sol (**3 pts**)-3 not same
    ○ This is all or nothing.
• Used the algorithm correctly (**6 pts**)
    ○ Uses Newton's Method (5/6 pts)
    ○ Accuracy is to 9 decimal places -1 this one is 8 decimal places
• Naming is correct (**1 pts**)
    ○ Source code " newton_cube **.c**"
    ○ Output file " newton_cube.**out**"
	6	10

Code Requirements		

• Tar ball MUST be named (2/10 pts)
cse122_fname_lname_hw0.tar.gz
• Must use a Makefile (2/10 pts)
• Must have doxygen comments (3/10 pts) -3 no doxygen comments
• Must have correct coding style (3/10 pts)
    ○ 8 space tabs
    ○ Proper bracket placement
    ○ etc.
	7	10

	83	**100**

CSE 122 – HW 0 Rubric

Good work overall! See above for where you lost points. The biggest issue was your outputs didn't diff properly with the .sol outputs. Other than that and your lack of doxygen comments, things looked great!