Comp 333 Homework #3 Fall 2020

( 20 pts) Due: Upload by Oct 20 at 5 pm on Canvas

**General Directions: Create a Dr Racket edit file called homework3.rkt . Put all your source code for the following problems in this file. Number the problems. Run the file in Dr Racket and use the interpreter to test your code. Put a copy of the interpreter screen showing the results of your run, as a comment at the bottom of your homework3.rkt file. Upload your single homework3.rkt file to Canvas. [Do not covert to a pdf or Word file. I plan to run your code in Dr Racket and so need you to upload a .rkt file.**

**FORMAT: Add your name, Comp 333, Homework #3 and Due Date: Oct 20 as a comment at the top of your homework3.rkt file. It must be embedded in the file. Put your problems in numeric order. Label as Problem 1, Problem 2, etc.**

**;Problem 1**

**<source code here>**

**;Problem 2**

**<source code here>**

**;Problem 3**

**<source code here>**

**;Problem 4**

**<source code here>**

**:Problem 5**

**<source code here>**

**; Test Case expressions and results (labeled)**

**<Put copy of interpreter screen showing tests and results here, in problem numeric order(labeled) >**

**For all problems, use only the Scheme functions and special forms we have studied in class or that are mentioned in the problem statement. You must use the same functions names and arguments as described; otherwise your code will fail my runtime tests.**

1. ( 3 pts) Write a Scheme function called range4 to find the difference between the smallest and largest of four numbers a,b,c,d. Hint: Use min and max. Your function should work for any 4 numbers in any order.

**Test your function range4 on the arguments 3 14 2 15 ; 1.5 4.6 3.1 – 7.2 ; 2 4 6 8**

1. ( 3 pts) Write an Scheme function called f with two arguments a and b that returns a useful error message if a and b are not both numbers; returns (a + b)/2 if a > b; returns 2 / (a + b) if a < b; returns a\*b if a = b. Use cond. Use number? to test if an argument is a number.

For example, ( f 5 8) returns 2/13 ; ( f 4.5 3.5) returns 4.0

**Test your function f on the arguments 10 6; 3 7 ; 4 4; 6.7 9.8; “red” 3**

1. ( 4 pts) Write two Scheme functions called second1 and third1 that return the second and third element of a list argument. Your function should check that the argument is a list and has length at least 2 for second and at least 3 for third. Return a useful error message if the list fails the test. Use list? to test for a list.

Example: (second1 ‘( a b c d e) ) returns ‘b

**Test both functions on each of the following arguments**

**‘( 1 7 4 9 10) ; ‘( apple orange) ; ‘( “red” “blue” “green”) ; ‘( ( a b c) ( d e f ) ( q w e r t y) )** ; **101**

**Note: There are 10 tests here.**

1. ( 6 pts) Write a Scheme function called triangleArea that uses Heron’s formula to determine the area of a triangle formed by three non-collinear points on the x-y plane. Each point is represented as a list of two numbers. Use let\* .

Example: (triangleArea '( 0 0) '(4.2 0) '(4.2 8) ) => 16.799999999999997

Write a helper function called distance the computes the distance between two given points in the x-y plane.

Example: > (distance '( 4.5 8.9 ) '( -2.1 100)) => 91.33876504529717

Use this function in the triangleArea function.

**Testcases: Use the distance function on the following arguments: '( 4.5 8.9 ) '( -2.1 100) ;**

**‘(-20.1 5.0) ‘( 30.2 -6.0)**

**Testcases: Use the triangleArea function on the following arguments:**

**‘( 0 0 ) ‘( 4.2 0) ‘( 4.2 8) ; ‘( 4.5 8.9 ) ‘( -2.1 100) ‘( 0 -5 ) ; ‘(1 1 ) ‘( 6 6) ‘(-6 6)**

1. (4 pts) Write a recursive Scheme function called sum1 to find the sum of the integers from 1 to n. Check that n >= 1 and check that n is an integer. Use integer? to check that n is an integer. Use cond. Do not use a sequence of nested ifs.

Example: ( sum1 200 ) returns 20100

Use following algorithm ( written in pseudocode here)

sum1(n)

if ( n is not an integer)

return a useful error message

else if ( n < 1)

return a useful error message

else if ( n = = 1)

return 1

else

return n + sum1( n – 1)

**Test sum1 on n = 10, 39, 100, - 5, 25.4**