CS 270 - Lab 6

M. Boady, B. Char, J. Johnson, G. Long, S. Earth

1 Introduction

You may work in teams of **one** or **two** students. Submit one copy for the entire group.

Write your answers on this lab sheet. Only what is written on this lab sheet will be graded.

This lab is due at the end of the class period. You may not continue to work on it once class has ended.

This lab contains 4 questions.

Grading

- 25 points - Putting everyones names on this page
- 20 points - Earned for each correct question (Answer is fully correct)
- 5 points - Earned for **partial credit** any question

No additional point amounts can be earned. You cannot earn 7 points on a question for example.

The maximum score for a lab is 100. If you get everything correct, that adds up to 105 points but will be reduced to 100.

A question will be marked correct as long as it covers all requirements of the question. It does not need to be perfect, but must be fully correct. A single typo or very minor issue where the intention is clear and all requirements are met would still earn full points.

We want you to complete questions fully, not try to earn partial credit on multiple questions. You may ask your Professor/Course assistant questions during lab.

Labs must be done in the presence of an instructor and/or course assistant or credit will not be given.

Partners should alternate each class day which person is physically typing and submitting the lab.

Do not split up the problems or you risk not finishing on time due to the cumulative nature of the questions.

Enter the name of the student in the group

Member 1 (submitter): Elan Rubin

Member 2: ______________________

Question 1 :

Define a Racket function (rem a b) that computes the remainder of a divided by b ($a \mod b$). Your implemenation must be recursive. You may not use built in remainder commands or math formulas to avoid recursion. Tthe goal of this exercise is to build it yourself. The below table shows a few values for the function.

You **must** solve this problem using recursion.

(rem 0 2)	0
(rem 1 2)	1
(rem 2 2)	0
(rem 3 2)	1
(rem 4 2)	0
(rem 9 3)	0
(rem 10 3)	1
(rem 9 4)	1
(rem 27 14)	13

The input and output contracts for this function are given below.

```
;input-contract: a is a nonnegative integer
;      and b is a positive integer.
;output-contract: an integer that is the remainder of a being divided by b
```

Give your function (rem a b) in Racket Syntax.

```
(define (rem a b)
  (if (< a b)
     a
     (rem (- a b) b)))
```

Question 2 :

Define a Racket function (R n) that sums a series of fractions. The below table shows the first few values for the function. You **must** solve this problem using recursion.

You may use the function (**expt** a b) for exponents.

(R 0)	0	Base Case
(R 1)	1/2	$0 + 1/2$
(R 2)	3/4	$0 + 1/2 + 1/4$
(R 3)	7/8	$0 + 1/2 + 1/4 + 1/8$
(R 4)	15/16	$0 + 1/2 + 1/4 + 1/8 + 1/16$
(R 5)	31/32	$0 + 1/2 + 1/4 + 1/8 + 1/16 + 1/32$

(a) Provide the Input and Output Contract for this function

; input-contract: n is a nonnegative integer number
; output-contract: the output is a number, representing the series sum

(b) Give your function (R n) in Racket Syntax.

```
(define (R n)
  (if (= n 0)
      0
      (+ (/ 1 (expt 2 n)) (R (- n 1)))))
```

Question 3 :

Define a Racket function (largestMultiple max mult) that finds the largest number that is less than or equal to `max` and is a multiple of `mult`. The below table shows the a few values for the function.

You **must** solve this problem using recursion.

(largestMultiple 0 7)	0
(largestMultiple 100 2)	100
(largestMultiple 100 7)	98
(largestMultiple 29 3)	27
(largestMultiple 200 9)	198
(largestMultiple 129 5)	125
(largestMultiple 4000 11)	3993

(a) Provide the Input and Output Contract for this function

; input-contract: both inputs (max and mult) are positive integer numbers
; output-contract: the output number is the integer that's the largest multiple of mult, >= max

(b) Give your function (largestMultiple max mult) in Racket Syntax.

```
(define (largestMultiple max mult)
  (if (< max mult)
      0
      (if (= (rem max mult) 0)
          max
          (largestMultiple (- max 1) mult))))
```

Question 4 :

Define a Racket function (logStar n) that determines how many times you need to apply ln before n is less than 1. The below table shows the a few values for the function.

You **must** solve this problem using recursion.

You may use the function (**log** a) to compute $\ln a$.

Example: We want to compute (logStar 50).

We apply $\ln 50 = 3.9120 \cdots$.

We apply $\ln \ln 50 = 1.364054 \cdots$.

We apply $\ln \ln \ln 50 = 0.31046 \cdots$.

We needed to apply ln three times to get below 1. Therefore (logStar 50) will return 3.

(logStar 1/2)	0
(logStar 1)	1
(logStar 10)	2
(logStar 50)	3
(logStar 100)	3
(logStar (expt 10 10))	4
(logStar (* 6 (expt 10 79)))	4

Note: It is estimated that there are $6 * 10^{79}$ atoms in the universe.

(a) Provide the Input and Output Contract for this function

; input-contract: n is a positive real number
; output-contract: a nonnegative integer number showing the amount of times ln() needs to be applied to the input n before the result is < 1

(b) Give your function (logStar n) in Racket Syntax.

```
(define (logStar n)
  (if (< n 1)
     0
     (+ 1 (logStar (log n)))))
```