CS 270 - Lab 1

B. Char, M. Boady, and J. Johnson

Week 1

1 Introduction

For this lab, you will work in a team of 1-3 students.

Assign Roles:

Each member of your group must be assigned a role. No member of the group can have the same position twice in a row.

Spokesperson: talks to the instructor, TA, and other teams. Only the spokesperson may raise their hand to ask a question.

Scribe: Records the team's answers on the Activity Sheet.

Developer: Writes and executes code. Uses computers to calculate answers. The developer is the only group member that may write code during lab. (If in a group of 2, this job is taken by the spokesperson.)

Enter the name of the student in each role

Spokesperson: _____

Scribe: _____

Developer: _____

Score (out of 100): _____ **Graded By**: _____

Question 1 : 15 points

2 Functional Notation

The Racket Programming Language is a purely functional language. It has a syntax that is different from most programming languages. Each command starts and ends with parenthesis. The function name is always the first item inside the parenthesis. The function name is followed by any input arguments.

Standard Notation	Racket Notation	Value
1 + 2	(+ 1 2)	3
2 * 5	(* 2 5)	10
2/3	(/ 2 3)	$\frac{2}{3}$
$7 < 9$	(< 7 9)	True (#t)
if(9 < 10){return 1;}else{return 2;}	(if (< 9 10) 1 2)	1

Write each of the below expressions in Racket Notation. Translate the expressions exactly, do no evaluate them.

(a) (3 points) 7 + 9 * 3

(b) (3 points) $90 < 100$

(c) (3 points) (50 * 3) > 100

(d) (3 points) if(2 - 5 > 0){ return 12;}else{return 15;}

(e) (3 points) 9 * 3 + 2 * 4

Question 2 : 15 points

What does each of the following expressions evaluate to?

(a) (3 points) (+ (* 9 (+ 1 1)) 2)

(b) (3 points) (- 9 10)

(c) (3 points) (* 9 (/ 7 9))

(d) (3 points) (if (< 1 2) 3 4)

(e) (3 points) (if (> 8 9) 12 15)

3 Function Definitions

Racket is designed for recursive functions. Any iteration must be completed recursively. The factorial function is the product of all numbers between 1 and n.

$$n! = 1 * 2 * 3 * 4 \cdots * (n-1) * n = \begin{cases} 1 & n = 0 \\ n * (n-1)! & n > 0 \end{cases} \quad (1)$$

The Racket definition is provided below.

```
(define (fact n)
  (if (<= n 1) 1 (* n (fact (- n 1))))
)
```

The first few return values are

(fact 0)	1
(fact 1)	1
(fact 2)	2
(fact 3)	6

Question 3 : 24 points

Answer each of the following questions.

(a) (3 points) What is value is returned by (fact (- 4 1))?

(b) (3 points) What value is return by (* 4 (fact (- 4 1)))?

(c) (3 points) Is (<= 4 1) true or false?

(d) (3 points) What is the return value of (fact 4)?

(e) (3 points) What is the return value of (fact 5)?

(f) (3 points) What is the return value of (fact 6)?

(g) (3 points) How many times does (<= n 1) return false when (fact 6) is executed?

(h) (3 points) How many times does (<= n 1) return true when (fact 6) is executed?

4 Recursive Thinking

Think about exponents. Powers of two are very common in Computer Science. By definition,

$$2^0 = 1 \tag{2}$$
$$x^0 = 1 \tag{3}$$

Question 4 : 21 points

Solve the below to derive a recursive formula for exponents.

(a) (3 points) What is the numerical value of 2^1?

(b) (3 points) Define 2^1 recursively in terms of 2 and 2^0

(c) (3 points) What is the numerical value of 2^6?

(d) (3 points) Write 2^6 recursively in terms of 2^5 and 2.

(e) (3 points) What is $\frac{2^n}{2^{n-1}}$?

(f) (6 points) Write a Racket function (pow a b) that computes a^b. You may assume $b \geq 0$.

5 Recursive Summation

Define a Racket function (S n) that sums up all numbers from 0 to n recursively. The below table shows the first few values for the function.

$S(0)$	0
$S(1)$	1
$S(2)$	3
$S(3)$	6
$S(4)$	10
$S(5)$	15

Question 5 : 20 points

Give your function (S n) in Racket Syntax.

Question 6 : 5 points

Group Discussion Question: Imagine you have to write a program to parse Racket functions. What do you think would be easier writing a parser for Racket functions or Python functions? Why?