## **CSE 240 Homework 11, Fall 2015 (50 points)**

Due Saturday, November 14, 2015 at 11:59PM, plus a 24-Hour grace period

### Introduction

The aim of this assignment is to make sure that you understand and are familiar with the concepts covered in the lectures. By the end of the assignment, you should have

* strong concept of functional paradigm.
* strong concept of pairs and list.
* understood the use of let-form and unnamed procedure.
* applied recursion to solve complex problems.

**Reading**: chapter 4 and course notes (slides).

You are expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their office hours.

**Preparation**: Complete the multiple choice questions in the textbook exercise section. The answer keys can be found in the course Web site. These exercises can help you prepare for your weekly quiz and the exam. You are encourage to read the other exercise questions and make sure you understand these questions in the textbook exercise section, which can help you better understand what materials are expected to understand after the lectures and homework on each chapter.

You are expected to do the majority of the assignment outside the class meetings. Should you need assistance, or have questions about the assignment, please contact the instructor or the TA during their office hours.

You are encouraged to ask and answer questions on the course discussion board. (However, **do not share your answers** in the course discussion board.)

### Programming Exercise (50 points)

In this assignment you will implement five new procedures. You may only use the list operations shown in the slides - not any of the additional library procedures in Scheme.

1. The Ackermann function is defined recursively for two nonnegative integers s and t as follows:

A(s, t) =

1.1 Follow the fantastic-four design steps to write a recursive procedure to implement the procedure (Ackermann s t). [6 points]

1.2 Use comments to indicate in the code: [4 points]

* + The size-n problem
  + All the stopping conditions and the return values.
  + All the size-m problems
  + The lines of code that construct the size-n solution from the size-m solutions.

Test cases:

(Ackermann 0 0) 🡪 1

(Ackermann 1 1) 🡪 3

(Ackermann 2 3) 🡪 9

(Ackermann 3 4) 🡪 125

(Ackermann 3 7) 🡪 1021

The full table of Ackermann values can be found at: <https://en.wikipedia.org/wiki/Ackermann_function>

Note, this is an extremely fast growing function for the first argument s. For a test case, such as (Ackermann 4 3), it will take too long to calculate.

1. Create a procedure called (**combine-four** lst1 lst2 lst3 lst4) that combines 4 lists into one list, in order of the given input lists. [4 points]

Test case: (**combine-four** '(1 2) '(3 4) ‘(5 6) ‘(7 8 9)) should give ‘(1 2 3 4 5 6 7 8 9)

1. Create a procedure that returns the last **n** items in the list **lst**.

3.1Use a **named** procedure called (**last-n** lst n) to implement task. [3 points]

Test case: (**last-n** '(1 6 3 4 5) 2) should give '(4 5)

3.2Use an **unnamed** procedure ((lambda (lst n) (. . .) '(1 6 3 4 5) 2) to implement task, where '(1 6 3 4 5) 2 are the arguments. [3 points]

The unnamed procedure should give '(4 5)

1. Create a procedure called (**first-n** lst) that returns the first **n** items in the list **lst**. **You must read in the value for n.** **You must not change the order of the list elements.**   [5 points]

 Test case with read-in input 3: (**first-n** '(1 5 3 4 5)) should give '(1 5 3)

1. Create a procedure called (**shuffle** lst1 lst2) that returns a shuffle of two lists. Shuffle the lists by alternating elements between the two lists. *You can*assume that both lists have the same length.  [10 points]

Test case: (**shuffle** ‘(1 2 3) ‘(a b c)) should give '(1 a 2 b 3 c).

1. Create a procedure called (**shufflePairs** one two) that returns a shuffle of two lists in pairs. *You can*assume that both lists have the same length.   [5 points]

Test case: (**shufflePairs**  ‘(1 2 3) ‘(a b c)) is '((1 . a) (2 . b) (3 . c)).

1. Create a procedure called (**addSquare** lst), where lst is a list of numbers. The procedure will calculate the square of each number in the list, add all the squared values, and then return the sum. If lst is empty, this procedure must return “empty list”. If lst is not empty, the procedure should call another procedure called (**addNonEmptyList** lst) that returns the sum of square values of all elements in the lst. If lst is empty, (**addNonEmptyList** lst) procedure should return 0 (the value a the stopping condition). For example, (**addSquare** ‘(1 2 3)) should return the sum of 12 + 22 + 32 [10 points]

Note, you could use map function, but you do not have to use it.

**Test cases:**

(addSquare '())

(addNonEmptyList '())

(addNonEmptyList '(1 2 3))

(addNonEmptyList '(1 4 3 2))

**They should return:**

"empty list"

0

14

30

\*\* You will submit this assignment in one definition file (.rkt).

### Grading of Programming Assignment

The TA will grade your program following these steps:

(1) The TA will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(2) Compile the code. If it does not compile, 20% of the points given in (1) will be deducted. For example, if you are given 20 points in step (1), your points will become 16 if the program fails to compile.

(3) If the code passes the compilation, the TA will execute and test the code. If, for any reason, the program gives an incorrect output or crashes for any input, 10% of the points given in (1) will be deducted.

### What to Submit?

You are required to submit your solutions in a compressed format (.zip). Zip all files into a single zip file. Make sure your compressed file is labeled correctly - lastname\_firstname.zip.

For this home assignment, the compressed file MUST contain the following:

Hw11.rkt (Scheme program)

No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded, even if the assignment is submitted late.

### Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

### Late submission deduction policy

* No penalty for late submissions that are received within 24 hours after the deadline;
* 10% grade deduction for every day it is late after the grace period;
* No late submission after Tuesday at 11:59PM.

### Academic Integrity and Honor Code.

You are encouraged to cooperate in study group on learning the course materials. However, you may not cooperate on preparing the individual assignments. Anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem. When you help your peers, you should never show your work to them. All assignment questions must be asked in the course discussion board. Asking assignment questions or making your assignment available in the public websites before the assignment due will be considered cheating.

The instructor and the TA will **CAREFULLY** check any possible proliferation or plagiarism. We will use the document/program comparison tools like MOSS (Measure Of Software Similarity: http://moss.stanford.edu/) to check any assignment that you submitted for grading. The Ira A. Fulton Schools of Engineering expect all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:

http://www.asu.edu/studentaffairs/studentlife/judicial/academic\_integrity.htm

ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.