Assignment 1

Due Monday October 13 at 1pm.
No late assignments will be accepted.

What to do

The questions below require you to write Scheme functions. The first few questions provide warm-up problems to get you used to the Scheme syntax, the programming environment and thinking like a functional programmer. In all questions, your Scheme functions should be well commented, and should be written in good functional programming style. In particular, do not use any functions or constructs that change the values of variables or have other side effects or that use a sequential processing of commands. To help you with this new way of thinking, I am disallowing the use of do, begin, or any function ending in !, such as set!, set-car!, vector-set!, etc. You must use recursion in all your solutions and use helper functions wherever appropriate. You may also find the following built-in Scheme functions useful: list, append, reverse, cadr and cddr. In general, simple solutions are preferred and will receive the most marks. To keep things simple, you may assume that the input to your functions is correct, so no error checking is required.

You should hand in four files: the source code of all your Scheme functions, a sample terminal session with the Scheme interpreter, the answers to the non-programming questions, and a scanned signed copy of the cover sheet at the end of the assignment. The source code should be well commented, and the terminal session should be short and should demonstrate that your functions work correctly. The files should be submitted electronically as described on the course web page. Be sure that we can run your Scheme code on the UTM computers. The terminal session should demonstrate that your functions work on all of the examples given in this assignment and on whatever other examples you deem necessary to demonstrate that your functions work correctly. (You will be graded on your choice of extra examples.)

Note: The marker has a limited amount of time for each assignment, so it is your responsibility to provide documentation and testing that allows him to quickly evaluate your work. As with all work in this course, 20% of the grade is for quality of presentation.

No more questions will be added
1. (5 points) Define a Scheme function \( \text{sumsq } N \) that returns the sum of the squares of the number from 1 to \( N \). For example, \( \text{sumsq } 4 \) => \( 1^2 + 2^2 + 3^2 + 4^2 = 30 \).

2. (7 points) Define a Scheme function \( \text{trim } N \ L \) that takes a list, \( L \), of arbitrary structure and a non-negative integer, \( N \), and returns a list in which the last \( N \) elements have been removed from \( L \). For example,

\[
(\text{trim } 4 \ '(1\ 2\ 3\ 4\ 5\ 6)) \Rightarrow (1\ 2)
\]

\[
(\text{trim } 2 \ '((1\ 2\ (3)\ 4\ 5\ (6))\ (7\ 8)\ (9\ 8)\ (7))) \Rightarrow ((1\ 2\ (3)\ 4\ 5\ (6))\ (7\ 8))
\]

3. (10 points) Define a Scheme function \( \text{mycount } A \ X \) which counts the number of times that symbol (or number) \( A \) occurs in nested list \( X \). For example,

\[
(\text{mycount } 1 \ ' (a\ 1\ b\ 2\ c\ 1)) \Rightarrow 2
\]

\[
(\text{mycount } \ 'a \ ' (a\ (b\ (a\ c)\ a)\ d)) \Rightarrow 3
\]

\[
(\text{mycount } ' a \ 'b) \Rightarrow 0
\]

\[
(\text{mycount } ' a \ 'a) \Rightarrow 1
\]

\[
(\text{mycount } ' a \ '()) \Rightarrow 0
\]

\[
(\text{mycount } 3 \ ' (((4))))) \Rightarrow 0
\]

4. (15 points total) Slow and fast computation of Fibonacci numbers.

(a) (5 points) Using non-linear recursion, define a Scheme function \( \text{slowfib } N \) that returns the \( N^{th} \) Fibonacci number, \( f_N \), for \( N \geq 0 \). Recall that \( f_0 = 0 \), \( f_1 = 1 \) and \( f_N = f_{N-1} + f_{N-2} \) for \( N \geq 2 \). Your definition of \( \text{slowfib} \) should be a straightforward encoding of the definition of \( f_N \), which will run very slowly for large values of \( N \). Do not use any helper functions or let expressions in your definition of \( \text{slowfib} \). Find a value of \( N \) for which \( \text{slowfib } N \) takes at least 5 minutes to finish.

(b) (10 points) Using linear recursion, define a Scheme function \( \text{fastfib } N \) that returns the \( N^{th} \) Fibonacci number for \( N \geq 0 \). \( \text{fastfib} \) should be much faster than \( \text{slowfib} \) and may use helper functions and let expressions. Print out the values of \( \text{fastfib } 100 \) and \( \text{fastfib } 1000 \).

Hint: Define a helper function, \( \text{fiblist } N \), that returns a list of the first \( N \) Fibonacci numbers. (This is a form of dynamic programming.)

5. (10 points) Define a Scheme function \( \text{leftmost } X \) that returns the leftmost symbol in a nested list \( X \). If \( X \) contains no symbols, then the empty list should be returned. For example,
6. (10 points) Define a Scheme function \texttt{myreplace A B X} that replaces all occurrences of symbol (or number) \texttt{A} by \texttt{B} in nested list \texttt{X}. For example,

\begin{verbatim}
(myreplace 'a 'd '(a b c a)) => (d b c d)
(myreplace '2 'd '(2 (3 (4 (5 2))))) => (d (3 (4 (5 d))))
(myreplace 'a 'c '(d e f)) => (d e f)
(myreplace 'a 'b 'a) => 'b
\end{verbatim}

7. (10 points) Let \texttt{P} be a predicate, \texttt{F} be a function, and \texttt{L} be a list. Define a Scheme function \texttt{mymap P F L} that replaces each element, \texttt{x}, of \texttt{L} by \texttt{(F x)} if \texttt{(P x)} is true, and leaves \texttt{x} unchanged otherwise. For example,

\begin{verbatim}
(mymap even? (lambda (X) (* X X)) '(1 2 3 4)) => (1 4 3 16)
(mymap number? (lambda (X) (+ 2 (* 3 X))) '(1 a 2 b 3 c)) => (5 a 8 b 11 c)
(mymap (lambda (X) (and (list? X) (not (null? X)))) cdr '(1 (a b c) 2 (d e) 3 (f))) => (1 (b c) 2 (e) 3 ()))
\end{verbatim}
8. (50 points) A road map consists of a set of cities and a set of roads connecting them. A road can be thought of as an unordered pair of cities. For example, here is a simple road map of an imaginary island with 10 cities and 14 roads:

Each road in the map is two way and represents a direct connection between cities. The distances between cities (in km and not drawn to scale) are given on the map.

Define a Scheme function \( \text{hamiltonian} \) \( \text{startcity} \) \( \text{endcity} \) \( \text{map} \) that returns a Hamiltonian path between \( \text{startcity} \) and \( \text{endcity} \), that is, a path that passes through every other city in \( \text{map} \) exactly once (no more and no less). Here, \( \text{map} \) is a Scheme expression representing the road map. The function should also return the total distance along the path. If such a path does not exist then return \#f or the empty list \( () \). Your function should work on any map, not just the one above. Demonstrate that your function works by testing it on other maps.

IMPORTANT: Your documentation should provide a high-level description of the algorithm and data structures used. In particular, be sure to clearly describe how you represent road maps.
NOTE: This is a difficult search problem. Start it early.

9. (10 points total) Consider the following Scheme function:

\[
\text{(define (f N) (if (zero? N) 0 (* 2 (+ 1 (f (- N 1)))))})
\]

Prove that this function has the following properties:

(a) \( f(N) = 6 + 4f(N-2) \) for all \( N \geq 2 \) (3 points)
(b) \( f(N) = 2^{N+1} - 2 \) for all \( N \geq 0 \) (7 points)

Justify each step of your proofs.

10. (15 points total) Consider the following two Scheme functions:

\[
\text{(define (f X)}
\begin{align*}
\text{(cond ((null? X) 0)} & \text{)}
\text{((number? X) 0)} & \text{)}
\text{((symbol? X) 0)} & \text{)}
\text{(#t (+ 1 (f (car X)) (f (cdr X)))))})
\end{align*}
\]

\[
\text{(define (g X Y)}
\begin{align*}
\text{(cond ((null? Y) X)} & \text{)}
\text{((symbol? Y) X)} & \text{)}
\text{((number? Y) X)} & \text{)}
\text{(#t (cons (g X (car Y)) (cdr Y))))})
\end{align*}
\]

(a) (4 points) In plain English, what do these functions do? Given some examples.

(b) (11 points) Prove that they have the following property:

\[
(f (g X Y)) = (f X) + (f Y)
\]

for all nested lists \( X \) and \( Y \). Justify each step of your proof.

No more questions will be added
Cover sheet for Assignment 1

Complete this page and submit it with your assignment.

Name: ______________________________
(Underline your last name)

Student number: ________________________

I declare that this assignment is solely my own work, and is in accordance with the University of Toronto Code of Behaviour on Academic Matters.

Signature: ______________________________