Problem 1

Write a procedure named each-matched-pair? that takes three inputs: lists \texttt{lst1} and \texttt{lst2} and a binary predicate called \texttt{pred?}.

The each-matched-pair? procedure applies \texttt{pred?} to each corresponding pair of members of \texttt{lst1} and \texttt{lst2}, e.g., if \texttt{lst1} is \texttt{'(1 2 3)} and \texttt{lst2} is \texttt{'(a b c)}, it evaluates \texttt{(pred? '1 'a)}, \texttt{(pred? '2 'b)}, and \texttt{(pred? '3 'c)}.

The each-matched-pair? procedure returns \texttt{#t} if \texttt{pred?} returns \texttt{#t} each and every time; otherwise it returns \texttt{#f}. If \texttt{lst1} and \texttt{lst2} have different lengths, each-matched-pair? ignores the extra members at the end of the longer list.

\begin{align*}
\text{(each-matched-pair? '(1 5 3 9) '(2 7 4 9) <=)} & \Rightarrow \texttt{#t} \\
\text{(each-matched-pair? '(1 7 3 9) '(2 5 4 9) <=)} & \Rightarrow \texttt{#f} \\
\text{(each-matched-pair? '(a b c) '() equal?)} & \Rightarrow \texttt{#t} \\
\text{(each-matched-pair? '() '(a b c) equal?)} & \Rightarrow \texttt{#t}
\end{align*}

Problem 2

Write a procedure named each-within-n? that takes three inputs: lists \texttt{lst1} and \texttt{lst2} and an integer \texttt{n}. It returns \texttt{#t} if each corresponding pair of members of \texttt{lst1} and \texttt{lst2} differ by \texttt{n} or less, regardless of which of the numbers is greater than the other. Otherwise, each-within-n? returns the Boolean value \texttt{#f}. You should use the procedure each-matched-pair? in your definition.

\begin{align*}
\text{(each-matched-pair? '(1 5 3 9) '(2 7 4 9) <=)} & \Rightarrow \texttt{#t} \\
\text{(each-matched-pair? '(1 7 3 9) '(2 5 4 9) <=)} & \Rightarrow \texttt{#f} \\
\text{(each-matched-pair? '(a b c) '() equal?)} & \Rightarrow \texttt{#t} \\
\text{(each-matched-pair? '() '(a b c) equal?)} & \Rightarrow \texttt{#t}
\end{align*}

Problem 3

Write a Scheme procedure called map-successive-pairs that takes as input a list called \texttt{lst} and a procedure called \texttt{fun} (which takes two arguments). The procedure map-successive-pairs applies \texttt{fun} to each successive pair of members of \texttt{lst}, i.e., it applies \texttt{fun} to the first and second members of \texttt{lst}, the second and third members, the third and fourth members, and so on. The procedure map-successive-pairs returns a list composed of the results of these applications of \texttt{fun}. The length of the output list will be one less than the length of the input list.

\begin{align*}
\text{(map-successive-pairs '(2 4 6 8) -)} & \Rightarrow \texttt{(-2 -2 -2)} \\
\text{(map-successive-pairs '(2 4 6 8) +)} & \Rightarrow \texttt{(6 10 14)}
\end{align*}

Problem 4

Write a Scheme procedure called uniformly-spaced? that takes a list of numbers as input and returns \texttt{#t} if each number on the list differs from the previous one by the same amount. If the
input list has fewer than two members, then \texttt{uniformly-spaced?} returns \texttt{#t}. Otherwise, it returns \texttt{#f}. You should use the procedures \texttt{map-successive-pairs} and \texttt{each-successive-pair?} in your definition.

\textit{Submitting}

Don't forget to submit your work using the \texttt{submit101} command!

\texttt{submit101 g-asmt05 asmt05}

(If the name of your directory is different from \texttt{asmt05}, change \texttt{asmt05} to whatever the name of your directory is.)