Objectives and Background

This MP is meant to give you practice writing and using higher order functions. In this assignment, you may not use any higher order functions provided by scheme, such as \textit{length} or \textit{map}. You may, however, write helper functions. Upon completion of this MP, you should understand:

- The basics of Scheme
- Recursive functions
- Higher order functions
- Induction

Problem 1

Write a hello world program in Scheme.

\texttt{> (hello\_world)}

"Hello world!"

Problem 2

Implement your own list length function. It should return the number of elements in the list.

\texttt{> (define mylist (list -1 0 1 2 3 ))}
\texttt{> (mylength mylist)}

5
Problem 3
Implement the Fibonacci function $fib(n)$ that runs in $O(n)$ time: that is, the computation time required to compute $fib(n)$ grows linearly with $n$. The Fibonacci function is defined as follows:

$$fib(n) := \begin{cases} 
0 & \text{if } n = 0; \\
1 & \text{if } n = 1; \\
fib(n - 1) + fib(n - 2) & \text{if } n > 1.
\end{cases}$$

> (fib 7)
13
> (fib 0)
0
> (fib 1)
1

Problem 4
Brownie Points Problem

Can you implement the Fibonacci function that runs in $O(\log(n))$ time?

Problem 5
Implement our own reverse function called myreverse. myreverse function should take a list of data, and reverse it.

> (myreverse (list 1 2 3 4 5))
(5 4 3 2 1)

Problem 6
Implement your own map function called mymap. mymap should take a function and a list as arguments, and apply the function to each element of the list, returning a list of results. So the $i^{th}$ item in the returned list is the result of applying the function to the $i^{th}$ item in the passed-in list.

> (mymap fib (list -1 0 1 2 3 7 ))
(0 0 1 1 2 13)

---

1Brownie Points are the kind of points that have no tangible value, but you still feel good when you get them. Besides, the solutions to these may be useful to you later in the course or in your CS career. To clarify, Brownie Point problems will not be graded and will have no effect on your grade.
Problem 7

Implement the funmap function. funmap should take a list of functions and a data list as arguments, and apply each function from the list of functions to the data list, returning a list of results. So the \(i^{th}\) item in the returned list is the result of applying the the \(i^{th}\) function to the data list.

\[
> \text{(funmap (list powerset mylength myreverse) (list 7 8 9))} \\
(((7 8 9) (7 8) (7 9) (7) (8 9) (8) (9) ()) 3 (9 8 7))
\]

Problem 8

Implement the powerset function. The powerset function takes a list of objects as an argument, and returns a list of all possible subsets of that list. The subsets themselves are represented as lists.

\[
> \text{(powerset (list 0 1 2))} \\
((0 1 2) (0 1) (0 2) (0) (1 2) (1) (2) (()))
\]

Problem 9

Brownie Points Problem

What is wrong with the following proof by induction?

**Theorem:** All horses are the same color.

**Proof:** by induction on the number \(n\) of horses.

**Base Case:** \(n = 1\). One horse is always the same color as itself.

**Induction Hypothesis:** Assume the statement is true for \(n - 1\) horses.

**Induction Step:** Consider a set \(S\) of \(n\) horses. Let \(a\) be one of the horses, and let \(S = S1 \cup S2\), where \(S1\) and \(S2\) each has \(n - 1\) horses and each contains horse \(a\). By the induction hypothesis, all horses in \(S1\) are the same color as horse \(a\) and all horses in \(S2\) are the same color as horse \(a\). So all horses in set \(S\) have the same color.

Handin

You should hand in a single file named ‘mp1.scm’ with the implementations of the above functions. The names of the functions and number and order of arguments of the functions
should be the same as in the problems.

Please see the CS 421 FAQ web page for handin instructions.