Before you can use the LISP interpreter, you need to add some lines to the \texttt{.bash\_profile} configuration file in your home directory. (Files starting with periods are hidden by default; to see the hidden files in a directory, use the \texttt{-a} option to \texttt{ls}). Add the following 2 lines after the last \texttt{PATH=} line toward the beginning of \texttt{.bash\_profile}:

\begin{verbatim}
PATH="\$PATH:/usr/sunshine/cmucl/bin"
MANPATH="\$MANPATH:/usr/sunshine/cmucl/man"
\end{verbatim}

Now type

\begin{verbatim}
source .bash\_profile
\end{verbatim}

from your home directory. This will reread the configuration file so the changes take effect. Now you are ready to use LISP. Just type \texttt{lisp} and you are on your way. At the \* prompt, type any well-formed LISP expression to see the result. When you are done, type the function \texttt{(quit)} to quit.

If you enter an incorrect lisp expression, the interpreter will put you into a debugging mode with a 0\} prompt. To return to the top level prompt, enter 0.

1. Write a LISP expression that replaces the second element in a list with a specified element. For instance, using \texttt{z} and \texttt{(a b c d)}, your expression should return \texttt{(a z c d)}

2. Write a LISP expression that adds 10 to the third element in a list of numbers and returns the resulting list. For example, if your list was \texttt{(4 8 12 3 11)}, your expression should return \texttt{(4 8 22 3 11)}

3. Write a LISP expression that concatenates two two-element lists. For example, concatenating \texttt{(a b)} and \texttt{(c d)} should return \texttt{(a b c d)}

4. Write a function \texttt{ends} that returns a list containing the first and last elements of the list passed in as a parameter. If the input list has less than two elements, return nil.

5. Write a function \texttt{rotate} that moves the first element of a list to the end of the list. (For example, \texttt{(rotate '(a b c d))} returns \texttt{(b c d a)}. Do not use iteration or recursion.

6. Write a predicate function \texttt{palp} that determines whether a list is a palindrome (reads the same backwards and forwards). Again, do not use iteration or recursion.

7. Write a recursive function \texttt{my-member} that searches a list for a particular element. This function should behave like the \texttt{member} predicate. But do \texttt{not} use \texttt{member}. (That would be too easy, now wouldn\'t it?)

8. How might you encode a binary tree in a LISP list?

9. Write a function that evaluates a fully parenthesized infix arithmetic expression. For example, \texttt{(+ 1 (* 2 3))} should return \texttt{7}.

10. Write a function that performs a depth first traversal of a binary tree. The function should return a list containing the tree nodes in the order they were visited.

11. Write a function that performs a breadth first traversal of a binary tree. The function should return a list containing the tree nodes in the order they were visited.

12. Write a function that determines whether an integer is prime. The function should return \texttt{t} or \texttt{nil}.

13. Write a function that returns a list of the first \texttt{n} primes, where \texttt{n} is the function parameter.

Work out the solutions to these problems using the LISP interpreter and hand in a typed copy of your solutions.