```
/**   File LISP-SYMBOLS.C **/

/** Symbol Tables and Installed Symbols **/
/* Include Files */
#include <stdio.h>
#include <string.h>
#include "lisp-header.h"

/** Variables **/
/* internal_symbols -- the symbol table for "Lisp" */
Hash_Table internal_symbols;

/* symbol_table -- pointer to the current symbol table */
Object *symbol_table;

/**     Predefined Internal Symbols   **/
# undef declare_symbol
#define declare_symbol(name, type)  \
        Object name
#include "int-lisp-syms.c"

/**   LISP FUNCTIONS   **/
/* INIT_HASH_TABLE -- set all hash buckets in a table to the empty list */
void init_hash_table (Hash_Table table)
 {
        int i;
        for (i=0; i < HASH_TABLE_SZ; i++)
                table [i] = NULL;
 }


/* HASH -- given a character string, return a hash code (from Aho, p. 436) */
int hash (char *str)
 {
        char *p;
        unsigned long g, h;
        /* from the book "Compliers" by Aho, Sethi & Ullman, p. 436 */
        h=0;
        for (p = str; *p != EOS; p++)
          {
                h = (h << 4) + (*p);            /*  left shift 4 bits         */
                g = h & 0xF0000000;                     /*  bitwise AND                     */
                if (g)
                  {
                        h = h ^ (g >> 24);              /*  bitwise exclusive OR */
```

```
                h = h ^ g;
            }
        }
    return ((int) (h % HASH_TABLE_SZ));
}

/* LOOKUP -- given a string, return symbol from 'symbol_table' or NULL */
Object lookup (char *str)
 {
    Object hash_bucket;                 /* list         */
    Object sym;                                 /* symbol    */

    hash_bucket = symbol_table [hash (str)];
    /*  walk linearly down 'hash_bucket' list looking for input string */
    while (hash_bucket != NULL)
      {
        sym = first (hash_bucket);
        if (strcmp (symbol (sym) -> print_name, str) == 0)
            return (sym);
        else
            hash_bucket = but_first (hash_bucket);
      }
    return (NULL);
}


/* INSTALL -- add a new symbol with given print string to 'symbol_table' */
Object install (char *str)
 {
    Object new_sym;
    int hash_index;

    new_sym = make_symbol (str);
    /* insert new symbol object at the front of appropriate hash bucket list */
    hash_index = hash (str);
    symbol_table [hash_index] = first_put (new_sym, symbol_table [hash_index]);
    return (new_sym);
}

/* INTERN -- return (possibly new and just installed) symbol of given name */
Object intern (char *str)
 {
    Object sym;                         /*  symbol */
    sym = lookup (str);
    if (sym == NULL)
        sym = install (str);
```

```
        return (sym);
 }


/* set_symbol_value -- set the value of an already installed symbol */
Object set_symbol_value (Object sym, Object val)
 {
        symbol (sym) -> value = val;
        return (val);
 }


/* install_with_value -- add a new symbol and its value to 'symbol_table' */
Object install_with_value (char *str, Object val)
 {
        Object new_sym;
        new_sym = install (str);
        set_symbol_value (new_sym, val);
        return (new_sym);
 }


/* install_internal_symbols -- set internal symbols known at compile time */
void install_internal_symbols (void)
 {
        symbol_table = internal_symbols;

        # undef declare_symbol
        #define declare_symbol(name,type)          \
          name = install_with_value (#name, type)
        #include "int-lisp-syms.h"

        install_with_value ("(", T_LPAREN);
        install_with_value (")", T_RPAREN);
 }
```