```
/* myhash.c -- symbol hashing table implementation -*- c -*- */
/*   Time-stamp: <mysymrechash.c (1998/08/18, 9:36:16) by alair a
t ime.usp.br)>   */
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <assert.h>
#include "mysymrec.h"
#ifdef CASE_INSENSITIVE
#define CASE(x) (toupper(x))
#else
#define CASE(x) (x)
#endif

static symrec **sym_table;
static int sym_table_size;

static unsigned
hash(char *key)
{
  unsigned register n = 0;
  /* this is not the best hash function but works */
  while (*key != 0)
    n = (n + n + CASE(*key++)) % sym_table_size;
  return n;
}

symrec *
putsym (sym_name,sym_type)
     char *sym_name;
     int sym_type;
{
  int h = hash(sym_name);
  symrec *ptr;
  ptr = (symrec *) malloc (sizeof (symrec));
  ptr->name = (char *) malloc (strlen (sym_name) + 1);
  strcpy (ptr->name,sym_name);
  ptr->type = sym_type;
  ptr->value.var = 0; /* set value to 0 even if fctn.  */
  ptr->next = sym_table[h];
  sym_table[h] = ptr;
  return ptr;
}
```

```
symrec *
getsym (sym_name)
     char *sym_name;
{
  symrec *ptr;
  for (ptr = sym_table[hash(sym_name)]; ptr != (symrec *) 0;
       ptr = (symrec *)ptr->next)
    if (strcmp (ptr->name,sym_name) == 0)
      return ptr;
  return 0;
}

void
create_sym_table(unsigned size)
{
  unsigned b;
  sym_table = (symrec **) malloc ( (sizeof(symrec)) * size );
  sym_table_size = size;
  assert (sym_table!=0);
  for (b = 0; b < size; b++)
    sym_table[b] = 0;
}

void
print_sym_table(void)
{
  unsigned b;  unsigned nel = 0, nbuc = 0;
  for (b = 0; b < sym_table_size; b++) {
    symrec *rec = sym_table[b];
    if (rec)
      nbuc++;
    for ( ; rec; rec = rec->next ) {
      printf ("%s\n", rec->name);
      nel++;
    }
  }
  fprintf (stderr, "%u buckets, %u nonempty (%u%%);"
    "%u entries, average chain %.1f.\n",
    sym_table_size, nbuc, 100 * nbuc / sym_table_size,  nel,
    nbuc ? nel / (double) nbuc : 0.0);
}
```