```
///////////////////////////////////////////////////////////////////////////////
////Class:         CS 445
////Semester:      Fall 2011
////Assignment:    Homework 4
////Author:        Dr. Robert Heckendorn, modified by Colby Blair
////File name:     symtab.h
////////////////////////////////////////////////////////////////////////////////

#ifndef _SYMTAB_H
#define _SYMTAB_H

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define false 0
#define true 1
#define bool int
#define DEBUG_PUSH 0x1
#define DEBUG_TABLE 0x2
#define DEBUG_LOOKUP 0x4
#define DEBUG_ALL 0xffffffff


typedef struct {
    char *name;
    char *type;
    char *scope;
    int depth;
    char* aux_flag;  //ie. 'const', etc
    void *ptr;
} SymTabEntry;


//
// Class SymTab
//
// A general simple stack of symbol tables that maps
// a char * to a void *.  Provides a user definable
// print routine for the objects stored in the symbol table.
// The print rouinte is defined when the constructor is called.
//
// debug flags setable by the debug method:
//   DEBUG_TABLE - announce entry to a scope and prints the symbol
//       table on exit from a scope.
//   DEBUG_PUSH - print everything that is pushed on the stack (uses
//       the print routine for printing the ptr value (treeNode *?)
// these flags are bit masks and so can be ored together to turn
// on multiple affects.  For example debug(DEBUG_TABLE | DEBUG_PUSH) would
// turn on both the DEBUG_PUSH and DEBUG_TABLE flags.
//
//  The four most important operations are insert, lookup, enter, leave.
//


void SymTab_init(void (* elemPrint)(void *));  // the constructor creates and sets t
he print routine
void SymTab_free();                            // destructor
void SymTab_debug(int newDebugValue);        // sets the debug flags
void SymTab_print();                         // prints the entire stack
bool SymTab_insert(char *, char*, char*, void *ptr);   // inserts a new ptr associat
ed with symbol sym
                                      // returns false if already defined
void *SymTab_lookup(char *sym);              // returns the ptr associated with sym
                                      // returns NULL if symbol not found
SymTabEntry *lookupSymTabEntry(char *sym);  // returns pointer to SymTabEntry associ
ated with sym
                                              // returns NULL if symbol not found
// scope functions
void SymTab_enter_scope(char *funcname);           // enter a function named funcname
bool SymTab_leave_scope();                         // leave that function
int SymTab_numEntries();                     // number of entries (more for debugging
)
```

```
int SymTab_depth();                               // depth of scopes on stack (useful in l
ater assignment)


#endif
```