11. Implement a C program to perform symbol table operations.

CODE:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define TABLE\_SIZE 100

typedef struct Symbol {

char name[50];

char type[50];

int scope;

struct Symbol\* next;

} Symbol;

Symbol\* symbolTable[TABLE\_SIZE];

unsigned int hash(const char\* name) {

unsigned int hash = 0;

while (\*name) {

hash = (hash << 5) + \*name++;

}

return hash % TABLE\_SIZE;

}

void insert(const char\* name, const char\* type, int scope) {

unsigned int index = hash(name);

Symbol\* newSymbol = (Symbol\*)malloc(sizeof(Symbol));

strcpy(newSymbol->name, name);

strcpy(newSymbol->type, type);

newSymbol->scope = scope;

newSymbol->next = symbolTable[index];

symbolTable[index] = newSymbol;

}

Symbol\* lookup(const char\* name) {

unsigned int index = hash(name);

Symbol\* entry = symbolTable[index];

while (entry != NULL) {

if (strcmp(entry->name, name) == 0) {

return entry;

}

entry = entry->next;

}

return NULL;

}

void display() {

for (int i = 0; i < TABLE\_SIZE; i++) {

if (symbolTable[i] != NULL) {

Symbol\* entry = symbolTable[i];

while (entry != NULL) {

printf("Name: %s, Type: %s, Scope: %d\n", entry->name, entry->type, entry->scope);

entry = entry->next;

}

}

}

}

int main() {

insert("x", "int", 1);

insert("y", "float", 1);

insert("func", "void", 0);

Symbol\* sym = lookup("x");

if (sym != NULL) {

printf("Found symbol: Name: %s, Type: %s, Scope: %d\n", sym->name, sym->type, sym->scope);

} else {

printf("Symbol not found.\n");

}

printf("Symbol Table:\n");

display();

return 0;

}

