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

Program:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define TABLE\_SIZE 10

typedef struct Entry {

char identifier[50];

char type[20];

struct Entry\* next;

} Entry;

Entry\* symbolTable[TABLE\_SIZE];

int hash(char\* identifier) {

int sum = 0;

for (int i = 0; identifier[i] != '\0'; i++) {

sum += identifier[i];

}

return sum % TABLE\_SIZE;

}

void insert(char\* identifier, char\* type) {

int index = hash(identifier);

Entry\* newEntry = (Entry\*)malloc(sizeof(Entry));

strcpy(newEntry->identifier, identifier);

strcpy(newEntry->type, type);

newEntry->next = symbolTable[index];

symbolTable[index] = newEntry;

printf("Inserted: %s, Type: %s at index %d\n", identifier, type, index);

}

Entry\* search(char\* identifier) {

int index = hash(identifier);

Entry\* current = symbolTable[index];

while (current) {

if (strcmp(current->identifier, identifier) == 0) {

return current;

}

current = current->next;

}

return NULL;

}

void display() {

printf("Symbol Table:\n");

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

printf("[%d]: ", i);

Entry\* current = symbolTable[i];

while (current) {

printf("(%s, %s) -> ", current->identifier, current->type);

current = current->next;

}

printf("NULL\n");

}

}

int main() {

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

symbolTable[i] = NULL;

}

insert("x", "int");

insert("y", "float");

insert("z", "char");

insert("count", "int");

insert("total", "double");

display();

char searchKey[50];

printf("\nEnter identifier to search: ");

scanf("%s", searchKey);

Entry\* found = search(searchKey);

if (found) {

printf("Found: %s, Type: %s\n", found->identifier, found->type);

} else {

printf("Identifier not found in symbol table.\n");

}

return 0;

}

Output:

