**Lab Expt - 5**

**Consider a super market scenario where sales manager wants to search for the customer details using a customer-id. Customer information like (custid, custname, & custphno) are stored as a structure, and custid will be used as hash key. Develop and execute a program in C using suitable data structures to implement the following operations:**

**a. Insertion of a new data entry.**

**b. Search for customer information using custid.**

**c. Display the records. (Demonstrate collision and its handling using linear probing method).**

#include <stdio.h>

#include <string.h>

#define TABLE\_SIZE 10 // Define the size of the hash table

// Structure to hold customer information

typedef struct {

int custid; // Customer ID

char custname[50]; // Customer Name

char custphno[15]; // Customer Phone Number

} Customer;

// Hash table

Customer hashTable[TABLE\_SIZE];

// Initialize the hash table

void initHashTable() {

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

hashTable[i].custid = -1; // -1 indicates an empty slot

}

}

// Hash function

int hashFunction(int custid) {

return custid % TABLE\_SIZE;

}

// Insert a new customer into the hash table

void insertCustomer(int custid, char \*custname, char \*custphno) {

int index = hashFunction(custid);

int originalIndex = index;

int i = 0;

// Linear probing in case of collision

while (hashTable[index].custid != -1) {

index = (originalIndex + i) % TABLE\_SIZE;

i++;

if (i == TABLE\_SIZE) {

printf("Hash table is full. Cannot insert new customer.\n");

return;

}

}

// Insert customer information at the found index

hashTable[index].custid = custid;

strcpy(hashTable[index].custname, custname);

strcpy(hashTable[index].custphno, custphno);

printf("Customer inserted at index %d\n", index);

}

// Search for a customer by custid

void searchCustomer(int custid) {

int index = hashFunction(custid);

int originalIndex = index;

int i = 0;

// Linear probing to find the customer

while (hashTable[index].custid != custid) {

index = (originalIndex + i) % TABLE\_SIZE;

i++;

if (i == TABLE\_SIZE || hashTable[index].custid == -1) {

printf("Customer with ID %d not found.\n", custid);

return;

}

}

// Customer found

printf("Customer found at index %d\n", index);

printf("Customer ID: %d\n", hashTable[index].custid);

printf("Customer Name: %s\n", hashTable[index].custname);

printf("Customer Phone Number: %s\n", hashTable[index].custphno);

}

// Display all records in the hash table

void displayHashTable() {

printf("Hash Table Contents:\n");

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

if (hashTable[i].custid != -1) {

printf("Index %d -> ID: %d, Name: %s, Phone: %s\n", i, hashTable[i].custid, hashTable[i].custname, hashTable[i].custphno);

} else {

printf("Index %d -> Empty\n", i);

}

}

}

// Main function

int main() {

int choice, custid;

char custname[50], custphno[15];

initHashTable();

while (1) {

printf("\nMenu:\n");

printf("1. Insert Customer\n");

printf("2. Search Customer\n");

printf("3. Display Records\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter Customer ID: ");

scanf("%d", &custid);

printf("Enter Customer Name: ");

scanf("%s", custname);

printf("Enter Customer Phone Number: ");

scanf("%s", custphno);

insertCustomer(custid, custname, custphno);

break;

case 2:

printf("Enter Customer ID to search: ");

scanf("%d", &custid);

searchCustomer(custid);

break;

case 3:

displayHashTable();

break;

case 4:

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}