WEEK 10 DS Lab

#include <stdio.h>

#include <stdlib.h>

#define MAX\_EMPLOYEES 100 // Maximum number of employees

#define HASH\_TABLE\_SIZE 7 // Size of the hash table

// Structure for employee record

struct Employee {

int key; // 4-digit key

// Other employee details can be added here

};

// Function prototypes

int hashFunction(int key);

void insertEmployee(struct Employee employees[], int hashTable[], struct Employee emp);

void displayHashTable(int hashTable[]);

int main()

{

struct Employee employees[MAX\_EMPLOYEES]; // Array to hold employee records

int hashTable[HASH\_TABLE\_SIZE] = {0}; // Hash table initialized with 0

int n, m, i;

// Input the number of employees

printf("Enter the number of employees: ");

scanf("%d", &n);

// Input employee records

printf("Enter employee records:\n");

for (i = 0; i < n; ++i)

{

printf("Employee %d:\n", i + 1);

printf("Enter key: ");

scanf("%d", &employees[i].key);

// Additional details can be input here

insertEmployee(employees, hashTable, employees[i]);

}

// Display the hash table

printf("\nHash Table:\n");

displayHashTable(hashTable);

return 0;

}

// Hash function: H(K) = K mod m

int hashFunction(int key)

{

return key % HASH\_TABLE\_SIZE;

}

// Function to insert an employee into the hash table

void insertEmployee(struct Employee employees[], int hashTable[], struct Employee emp)

{

int index = hashFunction(emp.key);

// Linear probing to resolve collisions

while (hashTable[index] != 0)

{

index = (index + 1) % HASH\_TABLE\_SIZE;

}

// Insert the employee key into the hash table

hashTable[index] = emp.key;

}

// Function to display the hash table

void displayHashTable(int hashTable[])

{

int i;

for (i = 0; i < HASH\_TABLE\_SIZE; ++i) {

printf("%d -> ", i);

if (hashTable[i] == 0) {

printf("Empty\n");

} else {

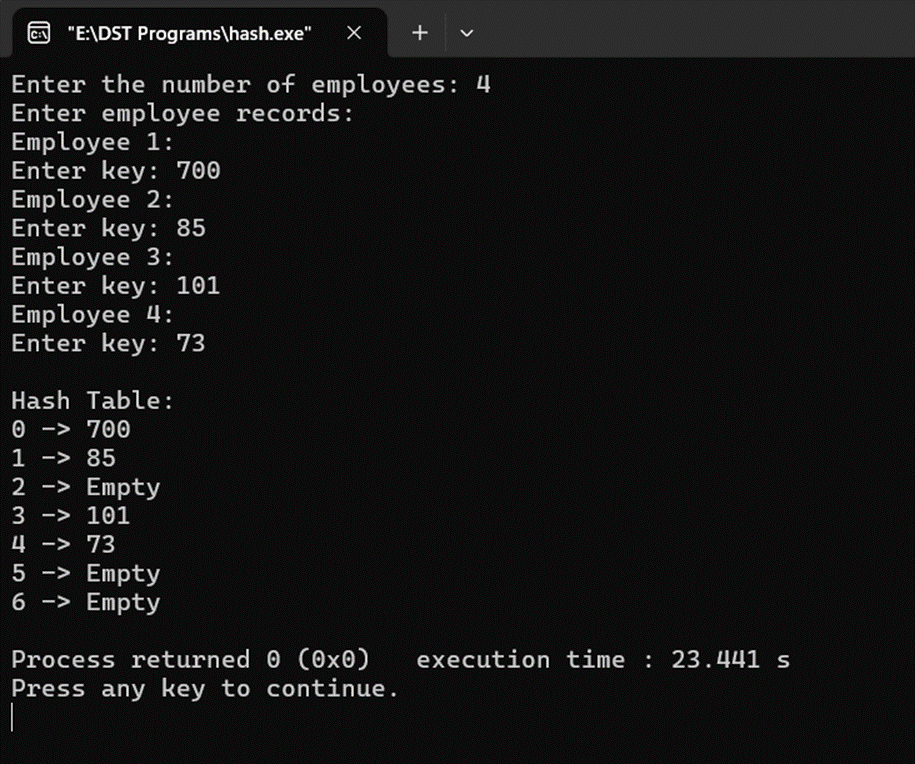
printf("%d\n", hashTable[i]);

}

}

}

**Output:**

****