Lab Practical Write-up

Aim:

To write a program to implement a hash table on names for a telephone directory in the C language.

Objective:

1. To understand the concept of hashing and its applications in data storage and retrieval.
2. To implement a hash table to efficiently store and search names in a telephone directory.
3. To explore different collision resolution techniques such as chaining and open addressing.
4. To analyze the efficiency of hash tables in comparison to other data structures for searching operations.

Theory:

A hash table is a data structure that maps keys to values using a hash function. This function computes an index in an array where the value will be stored. Hash tables provide quick insertion, deletion, and search operations, making them useful for applications like telephone directories.

Hash Function:

A hash function transforms input data (keys) into an index where the data is stored. A good hash function minimizes collisions (cases where multiple keys map to the same index).

Collision Resolution Techniques:

1. Chaining: Uses linked lists to store multiple values at a single index in case of collisions.
2. Open Addressing: Finds an alternative index using probing techniques such as:
   - Linear Probing
   - Quadratic Probing
   - Double Hashing

Advantages of Hash Tables:

- Fast lookup, insertion, and deletion (O(1) in best cases).
- Efficient memory usage for large datasets.
- Useful for applications requiring quick access to records (e.g., telephone directories).

Disadvantages of Hash Tables:

- Collisions can reduce efficiency, leading to O(n) operations in worst cases.
- Requires a well-designed hash function for optimal performance.
- Open addressing techniques can lead to clustering and performance degradation.

Example:

Consider a telephone directory with names and corresponding phone numbers. The hash function converts each name into an index for storing and retrieving data efficiently.

Sample Input:

Enter number of entries: 3

Enter name and phone number:

John 9876543210

Alice 9123456789

Bob 8901234567

Sample Output:

Telephone Directory:

John -> 9876543210

Alice -> 9123456789

Bob -> 8901234567

Conclusion:

Hash tables provide an efficient way to store and retrieve telephone directory data. They offer fast search operations compared to linked lists and arrays. However, their performance depends on the quality of the hash function and collision resolution method used. Understanding these concepts is crucial for designing efficient data structures in real-world applications.