**Aim:** Hashing Techniques.

**Objectives:** The main objective of this assignment is to understand and implement various hashing techniques such as modulo division, mid-square, digit extraction, fold shift and fold boundary in C++. The goal is to understand how different hashing algorithms can be used to store the data in a specific location.

**Tools Used:** VS Code C++.

**Concept:**

**Problem Statement**:

1.) Implement modulo division hash function.

2.) Implement mid-square hash function.

3.) Implement digit extraction hash function.

4.) Implement fold shift hash function.

5.) Implement fold boundary hash function.

**Solution**:

**Observation**: In this practical session I learned about different sorting techniques such as bubble sort, insertion sort, selection sort, quick sort, radix sort, merge sort.

Bubble Sort: Simple but inefficient for large datasets; repeatedly swaps adjacent elements. Best for small or simple tasks.

Insertion Sort: Builds a sorted array one element at a time; efficient for small or nearly sorted datasets. Slower for large unsorted arrays.

Selection Sort: Finds the minimum element and places it at the beginning; easy to understand but inefficient for larger arrays. Best for small datasets.

Quick Sort: It works by selecting a "pivot" element and partitioning the array into two halves: one with elements less than the pivot and one with elements greater than it. This process is repeated recursively to sort the given array, making it particularly effective for large datasets.

Radix Sort: Sorts numbers digit by digit; non-comparative and efficient for integers. It is not suitable for -ve nos.

Merge Sort: Divides and conquers by merging sorted subarrays; it is much more faster but requires extra space. Great for large datasets.