**ASSIGNMENT NO. 12**

**Title:** Quick Sort

**Problem Statement:**

Write a C++ program to store first year percentage of student in array of floating point numbers in ascending order using quick sort and display top 5 scorer.

**Learning Objective:**

Learning and implementation of quick sort algorithm.

**Pre-requisite:**

Sorting

Algorithm for quick sort

**Theory:**

A] Quick Sort

1. Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays.

2. A large array is partitioned into two arrays one of which holds values smaller than specified value say pivot based on which the partition is made and another array holds values greater than pivot value.

3. The quick sort partitions an array and then calls itself recursively twice to sort the resulting two subarray.

B] Advantages of Quick Sort:

1. The efficient average case of quick sort algorithm when compared to any sort algorithm is very high.

2. The quick sort produces the most effective and widely used method of sorting a list of any item size.

C] Disadvantages of Quick Sort:

1. A priority picked pivot can lead to bad runtimes.

2. It is difficult to implement the partitioning algorithm.

3. The average efficiency for the worst case scenario, which is not offset by the difficult implementation.

D] Complexity:

This algorithm is quite efficient for large sized data sets as its average and worst case complexity are of O(nlogn) where n are no. of items.

E] Applications of Quick Sort:

1. Used to sort large data.

2. Used for implementation in example for real life.

* **Algorithm:**

**Step 1 –** Start

**Step 2** − Choose the highest index value has pivot

**Step 3** − Take two variables to point left and right of the list excluding pivot

**Step 4** − left points to the low index

**Step 5** − right points to the high

**Step 6** − while value at left is less than pivot move right

**Step 7** − while value at right is greater than pivot move left

**Step 8** − if both step 5 and step 6 does not match swap left and right

**Step 9** − if left ≥ right, the point where they met is new pivot

**Step 10** - Stop