**DAY 2 – PRACTICE PROBLEMS for STUDENTS**

1. Apply Heap Sort to sort a Given set of marks of students in ascending order.
2. Given a string S with repeated characters. The task is to rearrange characters in a string such that no two adjacent characters are the same.
3. Create a sorting algorithm that minimizes the number of comparisons and swaps needed to sort an array of strings in lexicographic order usnig c++.

*Hint – You can use Radix Sort algorithm. Radix Sort is a non-comparative sorting algorithm that sorts elements by examining their individual digits or characters.*

1. Develop a sorting algorithm that efficiently sorts a large array of integers in ascending order.
2. Implement a sorting algorithm that works efficiently on a sparse matrix, sorting the rows in ascending order based on a specific column.
3. Develop a sorting algorithm that can handle arrays of floating-point numbers and sorts them in descending order.
4. Design a stable sorting algorithm that can efficiently sort a large collection of objects based on multiple criteria.
5. In a Given dictionary (English), check for the index of a given word in the same. Also in case the Word is not found Show appropriate message. Implementation using Binary Search.

***More practice Problems***

1. Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.

Consider the following dictionary

{ i, like, sam, sung, samsung, mobile, ice,

cream, icecream, man, go, mango}

Input: ilike

Output: Yes

The string can be segmented as "i like".

Input: ilikesamsung

Output: Yes

The string can be segmented as "i like samsung"

or "i like sam sung".

1. Given the heights of **N** towers and a value of **K**, Either increase or decrease the height of every tower by **K** (only once) where **K** > 0. After modifications, the task is to minimize the difference between the heights of the longest and the shortest tower and output its difference.

**Examples:**

***Input:****arr[] = {1, 15, 10}, k = 6****Output:****Maximum difference is 5.****Explanation:****Change 1 to 7, 15 to 9 and 10 to 4. Maximum difference is 5 (between 4 and 9). We can’t get a lower difference.*

***Input:****arr[] = {1, 5, 15, 10}, k = 3****Output:****Maximum difference is 8, arr[] = {4, 8, 12, 7}*