**National University of Computer & Emerging Sciences**

**Karachi Campus**



**Data Structure**

**Section: BAI-4A**

**Group Members:**

**Abdul Rehman Nazeer 22K4078**

**Project Report**

**Introduction:**

* Our project takes any unsorted words file, then sort it efficiently.
* After that it will display some input formats.
* According to these five formats user will enter the word which he wants to guess, but here comes our main focus of our work:
* The word he will enter will not be completed ,as our input format says and our project will finds the all words completely which user wants to search with the help of trees.

**Working:**

Trie is an efficient information re**Trie**val data structure. Using Trie, search complexities can be brought to optimal limit (key length). If we store keys in binary search tree, a well balanced BST will need time proportional to **M \* log N**, where M is maximum string length and N is number of keys in tree. Using Trie, we can search the key in O(L) time. However the penalty is on Trie storage requirements.

**TOOLS USED:**

* Language:- cpp
* Compiler:- dev c++
* Libararies:-
* #include <bits/stdc++.h> common shorthand to include most of the standard C++ libraries
* #include<conio.h>
* TREE:- trie
* Sorting:- MERGE SORT time complexity of O(n log n)
* SEARCHING:- TRIE

**WHY C++?**

* C++ is more suitable for large scale software projects such as management system which includes multi-task, multi-factor and multi requirement.
* By using encapsulation, we can encircle the process and data so that each functional module is independent of each other
* We could then successfully restrict the modification and maintenance of the program to the class, which ultimately makes thing easier.
* By applying inheritance, we could share attributes from parent classes to improve code’s reusability.

By making use of polymorphism, it provides a way to abstract the complicated relationships

**PROBLEMS FACED:**

* Had so little time bcz of multiple projects, lack of coherency
* Confusion in distribution of work
* Time complexity

**How we counter:**

We gather at one place and discuss our contribution of work, we take help from book: “Let us C++” for the choice of classes and streams, we use sorted file because of time complexity. but after facing some difficulties we achieve our project.

**REAL LIFE APPLICATION:**

**AUTO COMPLETE:**

The most common use of the Trie Data structure is autocomplete. It has a wide range of uses, and its popularity is growing rapidly in this age of smartphones and smart devices. It's a process in which the device software uses the string prefix to guess the rest of a word a user is typing. Users can select one of the several options with tab gaps in Graphical User Interfaces.

**SPELL CHECKER:**

Trie data structure is used to store the data dictionary and algorithms for searching the words from the dictionary and provide the list of valid words for suggestion can be constructed.

**Browser history:**

Web browsers keep track of history of websites visited by the user.  
By storing this history, with number of visits to the specific websites as a key value, and organizing it on Trie data structure, the user is given suggestions of the website to visit when the prefix of the previously visited URL is written in the address bar.

**How a Trie Works?**

• A trie is a multi-level tree where each node represents a character (or part of a key).

• The root node represents an empty string, and each edge corresponds to a character in the strings you're storing.

• Each path from the root to a node represents a sequence or word.

• Tries are useful because common prefixes are shared among words, making them memory-efficient for certain tasks.