SYMBIOSIS INSTITUTE OF TECHNOLOGY

(Established under section 3 of the UGC Act 1956) Re - accredited by NAAC with 'A++' Grade

Course Title: CPPDS

Branch: E&TC

Batch: A

Subject: String Sorting using Linked List

Date of Submission: 18th October, 2023

Submitted to: **Prof. Parag Narkhede**

Submitted by:

i. **Arjunsingh J. Gautam -> 22070123043**
ii. **Ashish Balsure -> 22070123030**
iii. **Bhoumik Sundaram ->22070123034**

Project Documentation: String Sorting using Linked List

Table of Contents

1. Introduction

This project presents a C++ implementation of a linked list-based string sorting system. The project provides functionality to create a linked list, insert elements at the head or tail, delete elements from the head or tail, sort the linked list in ascending order, find repeated words, and count the occurrences of specific words in the list.

2. Project Overview

The project consists of a **Linked List** namespace containing functions for linked list operations. It includes features such as inserting and deleting nodes, sorting the linked list, finding repeated words, and counting word occurrences. The sorting algorithm used is Bubble Sort.

3. Project Structure

main.cpp: Contains the main function demonstrating the usage of linked list functions.

LinkedList.h: Header file containing declarations and definitions of linked list functions.

Node: A structure representing a node in the linked list.

4. Functions

*print_ll(Node head)**: Prints the linked list.

*insert_at_head(Node head, std::string val)**: Inserts a node at the head.

*insert_at_tail(Node head, std::string val)**: Inserts a node at the tail.

*delete_at_head(Node head)**: Deletes a node from the head.

*delete_at_tail(Node head)**: Deletes a node from the tail.

*length(Node head)**: Returns the length of the linked list.

*is_member(Node node, std::string search_value)**: Checks if a value is in the linked list.

*count_matches(Node head, std::string count_value)**: Counts the occurrences of a value in the linked list.

*repeated_words(Node head)**: Prints the repeated words in the linked list.

*sort_ll(Node head)**: Sorts the linked list in ascending order using Bubble Sort.

5. How to Use

Include "LinkedList.h" in your C++ file.

Create a Node* variable as the head of the linked list.

Use the provided functions to perform operations on the linked list.

6. Performance Analysis

The sorting algorithm used is Bubble Sort, which has a time complexity of O(n^2) in the worst case. This may not be the most efficient sorting algorithm for large datasets. Consider using more efficient sorting algorithms such as QuickSort or MergeSort for larger datasets.

7. Conclusion

This project provides a basic implementation of string sorting using a linked list. While it serves as a learning resource, it is important to consider more efficient algorithms for real-world applications with large datasets.

8. Contributors

Ashish Balsure	**-> 22070123030**
Bhoumik Sundaram	**-> 22070123034**
Arjunsingh Gautam	**-> 22070123043**