**A Combined Use of Linked List and Binary Tree**

**Group Number: Team Honor 3  
Team Members: Yankui Lee, Hengli Jia**

**Introduction**

**Our project combines two data structures—linked lists and binary trees—to construct a Huffman tree. This tree structure is used to efficiently compress text by following a rule that minimizes storage space.**

**Data Structures, Structure Chart, and Class Diagram**

**Details of the data structures, structure chart, and class diagram are provided in the PDF document.**

**Sample Output Screen**

**A sample output screen is included in the JDP document.**

**Major Debug Problems**

**First Problem**

**The initial challenge we faced was not fully understanding how to structure the linked list. We rushed into implementation without a clear plan, resulting in a disorganized structure. This lack of preparation forced us to rebuild parts of the project, which consumed a significant amount of time.**

**Lesson Learned:  
To prevent similar issues in future projects, we should begin with a structure chart, a data structure diagram, and a class diagram. These visual tools help clarify the organization and implementation strategies for large projects, ensuring a smoother development process.**

**Second Problem**

**The second major issue involved a recursive dependency between Huffman.h and LinkedList.h. Specifically, Huffman.h required the LinkedList structure, while LinkedList.h also depended on elements from Huffman.h. This circular dependency resulted in compiler errors because C++ does not allow circular inclusions.**

**While forward declarations offer a temporary fix, the better long-term solution is to rethink the design. Instead of both headers relying on each other, we should redesign the structures to establish a clear containment relationship. For instance, one structure can contain the other, removing the need for mutual dependencies. This not only resolves the circular inclusion issue but also simplifies the design of the main function.**