**JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY SEC-128, NOIDA**

**BATCH: F6**

**HASH TABLE-BASED DICTIONARY (SYNOPSIS)**



**SUBMITTED BY-**

Ankita Jha 9922103166

Khushi Singh 9922103148

Shivam Pandey 9922103165

Pragya Sharma 9922103161

**SUBMITTED TO-**

Dr. Ambalika Sarkar

**Acknowledgement**

We would like to express our sincere gratitude to everyone who has contributed to the successful completion of this project.

First and foremost, we would like to thank our project supervisor, Dr. Ambalika Sarkar, for providing us with valuable guidance and support throughout the project. Her insightful feedback and suggestions helped us to stay on track and achieve our objectives.

We would also like to extend our appreciation to our team members, for their hard work, dedication, and cooperation throughout the project. Each member played a crucial role in the project's success and made significant contributions towards achieving our goals.

Thank you all for your contributions to this project.

**Overview**

The Hash Table-based Dictionary project involves creating a dictionary application using C++ and various data structures to efficiently store and retrieve words and their meanings. The primary data structure used in this project is a hash table, and collision handling techniques like chaining (using linked lists) will be applied.

**Key Concepts and Data Structures Used**

**1. Hashing**: Hashing is the process of converting a key (word) into an index in the hash table. In this project, we'll use a hash function to generate these indices. A good hash function should distribute the keys evenly across the table to minimize collisions.

**2. Hash Table:** A hash table is an array-like data structure where each index corresponds to a "bucket" that can hold multiple key-value pairs. In C++, you can implement a hash table using an array of linked lists (for chaining).

**3. Chaining:** Chaining is a collision resolution technique that involves maintaining a linked list at each bucket in the hash table. If multiple keys hash to the same index, they are stored in the corresponding linked list.

**4. Linked List:** Linked lists are used to implement the chains in the hash table. In C++, you can create a linked list using a class with a pointer to the next node.

**Project Steps**

**1. Hash Function Design:** Design a hash function that can generate unique indices for words while minimizing collisions. The function should take a word as input and return an index in the hash table.

**2. Hash Table Implementation:** Create a hash table in C++ using an array of linked lists. Each linked list will store key-value pairs (word-meaning pairs) that hash to the same index.

**3. Adding Words:** Implement a function to add new words and their meanings to the dictionary. The hash function will determine the index where the word should be inserted, and chaining will handle collisions.

**4. Searching for Words:** Develop a function to search for a word in the dictionary. The hash function will identify the index, and you can traverse the linked list to find the word's meaning.

**5. Updating and Deleting Words:** Implement functions to update and delete words and their meanings from the dictionary. These functions should also consider collisions when accessing and modifying the linked lists.

**6. User Interface:** Create a user-friendly command-line interface (CLI) or graphical user interface (GUI) to interact with the dictionary. Users should be able to add, search, update, and delete words easily.

**7. Testing:** Thoroughly test the dictionary application by adding a variety of words and ensuring that searching, updating, and deleting functions work as expected. Test for performance by adding a large number of words.

**8. Collision Handling:** Ensure that collision handling using chaining is effective and that it doesn't cause performance issues even with a large number of entries.

**Additional Considerations**

**- Load Factor:** You can implement a load factor mechanism to dynamically resize the hash table when it becomes too full, ensuring efficient storage and retrieval.

**- Serialization:** Consider adding functionality to save and load the dictionary data from a file, allowing users to persist their dictionary between sessions.

**-Coding platform used:** VS code

**Conclusion**

This project will help you gain a deep understanding of hash tables, collision resolution techniques, and linked lists in the context of a real-world application. It's a valuable exercise for anyone looking to strengthen their C++ and data structure skills.