KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

COURSE: DATA STRUCTURES THROUGH C

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

PROJECT TITLE: PHONE DIRECTORY SYSTEM BY USING ROOBIN HOOD HASHING

TEAM NO : 4

TEAM MEMBERS:

24-5U6 (V.KEERTHAN)

24-5V7 (Y.SANJAY)

25-511 (PRAVEEN)

MENTOR: MURTHY SIR

PROBLEM STATEMENT

In a traditional phone directory, storing and searching for contact information (such as a person's name and phone number) can become inefficient as the number of records increases. Linear probing or basic hashing methods may lead to clustering and uneven search times, making lookups slower for certain entries.

To overcome this problem, Robin Hood Hashing can be implemented — an advanced open addressing technique that ensures fair distribution of search times by minimizing variance in probe lengths.

INTRODUCTION

The Phone Directory System is a data management application designed to store, search, update, and delete contact details such as names, phone numbers, and addresses efficiently. To achieve fast data retrieval and uniform storage, this system uses a hashing technique — specifically, Robin Hood Hashing.

Robin Hood Hashing is an advanced method of open addressing in hash tables. Unlike traditional linear probing, it minimizes clustering by ensuring that every record gets a fair chance to be close to its ideal position (home index). When inserting a new entry, if a collision occurs, the algorithm compares the probe distances and swaps elements so that entries that have traveled farther get priority. This results in a more balanced distribution and faster average search times.

OBJECTIVES

- To implement an efficient hashing mechanism for storing contacts.
- To use Robin Hood Hashing to reduce clustering and improve lookup performance.
- To perform operations such as:
    - Adding a new contact (name and phone number)
    - Searching for a contact by name or number
    - Updating contact information

- Deleting a contact
- Displaying all stored contacts

EXPECTED OUTPUT

- Efficient insertion and retrieval of contacts with minimized variance in search time.
- Display of probe count and average search time for performance analysis.
- A user-friendly console-based interface to manage the phone directory.

CONSTRAINTS

The Phone Directory System using Robin Hood Hashing has certain constraints that affect its performance and efficiency. One major limitation is the fixed size of the hash table, which restricts the number of contacts that can be stored; once it is full, rehashing is required, which consumes additional time and memory

SYSTEM ANALYSIS

The Phone Directory System is designed to efficiently store, search, and manage contact information such as names and phone numbers. It uses Robin Hood Hashing, an advanced hashing technique, to minimize search time and evenly distribute data in the hash table. The system analyzes user input to insert, delete, or search for contacts based on hash values.

SYSTEM DESIGN

The System Design of the Phone Directory System focuses on organizing how data is stored, accessed, and managed efficiently. The system uses a hash table as the main data structure, where each contact's name or phone number is converted into a hash value to determine its storage location. The Robin Hood Hashing technique is applied to reduce search time and balance probe lengths by swapping elements when needed. The design includes key modules such as Insertion, Search, Deletion, and Display to handle all operations on contacts

ALGORITHMIC DESIGN

Step 1: Start

Step 2: Initialize Hash Table

- Create an empty hash table to store contact details (Name, Phone Number, etc.).

Step 3: Compute Hash Value

- Use a hash function to find the index for each contact:
  index = hash(name) % table_size

Step 4: Insert Contact (Robin Hood Hashing)

1. If the position is empty → insert the contact.
2. If occupied → compare the probe count (distance from original position).
3. If the new contact has a higher probe count, swap them (Robin Hood principle).
4. Continue probing to the next slot until inserted.

Step 5: Search Contact

1. Compute hash value of the name.
2. Probe sequentially (using same rule) until contact is found or empty slot reached.

Step 6: Delete Contact (optional)

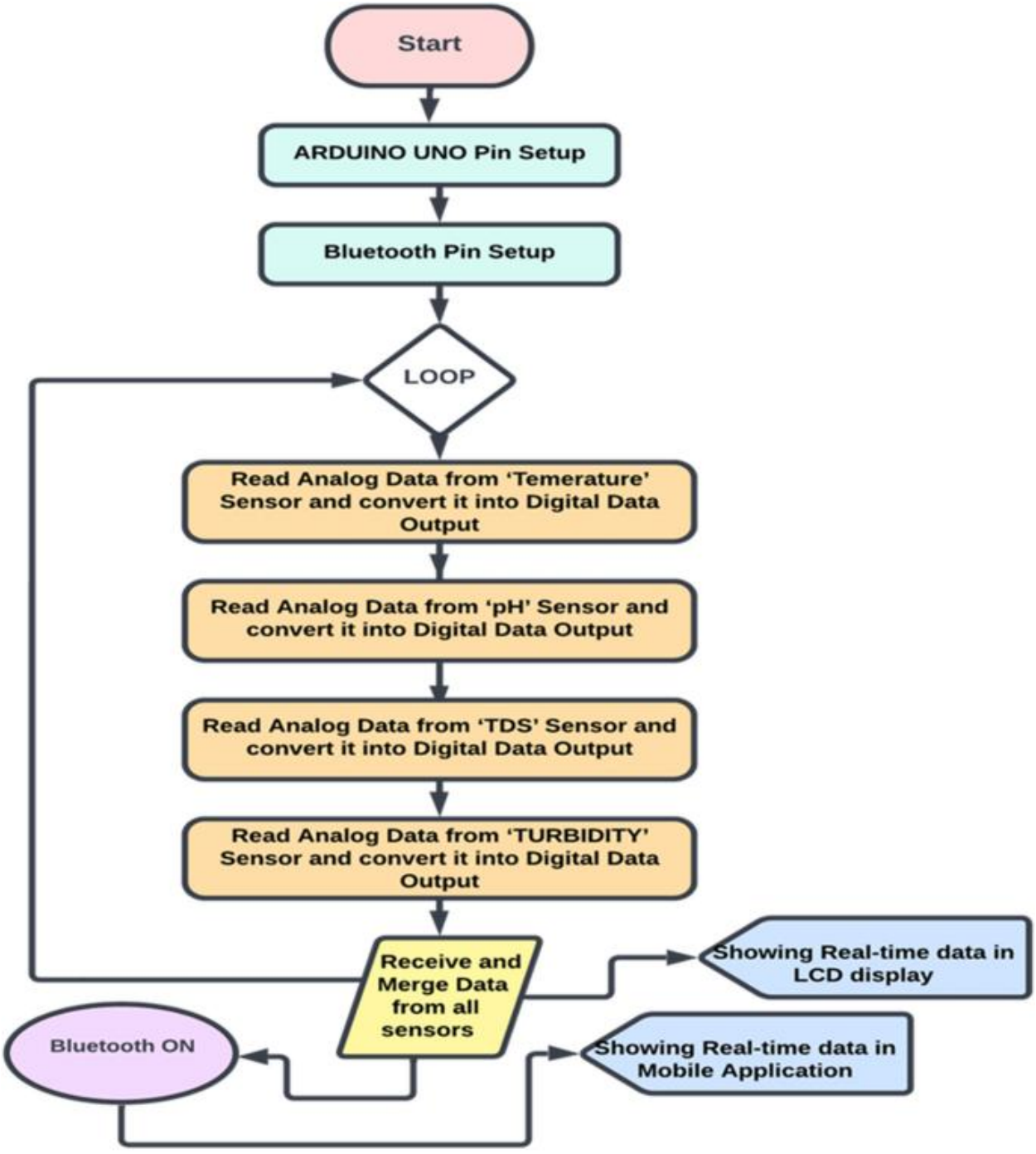
- Mark the slot as deleted or rehash elements to maintain order.

Step 7: Display All Contacts

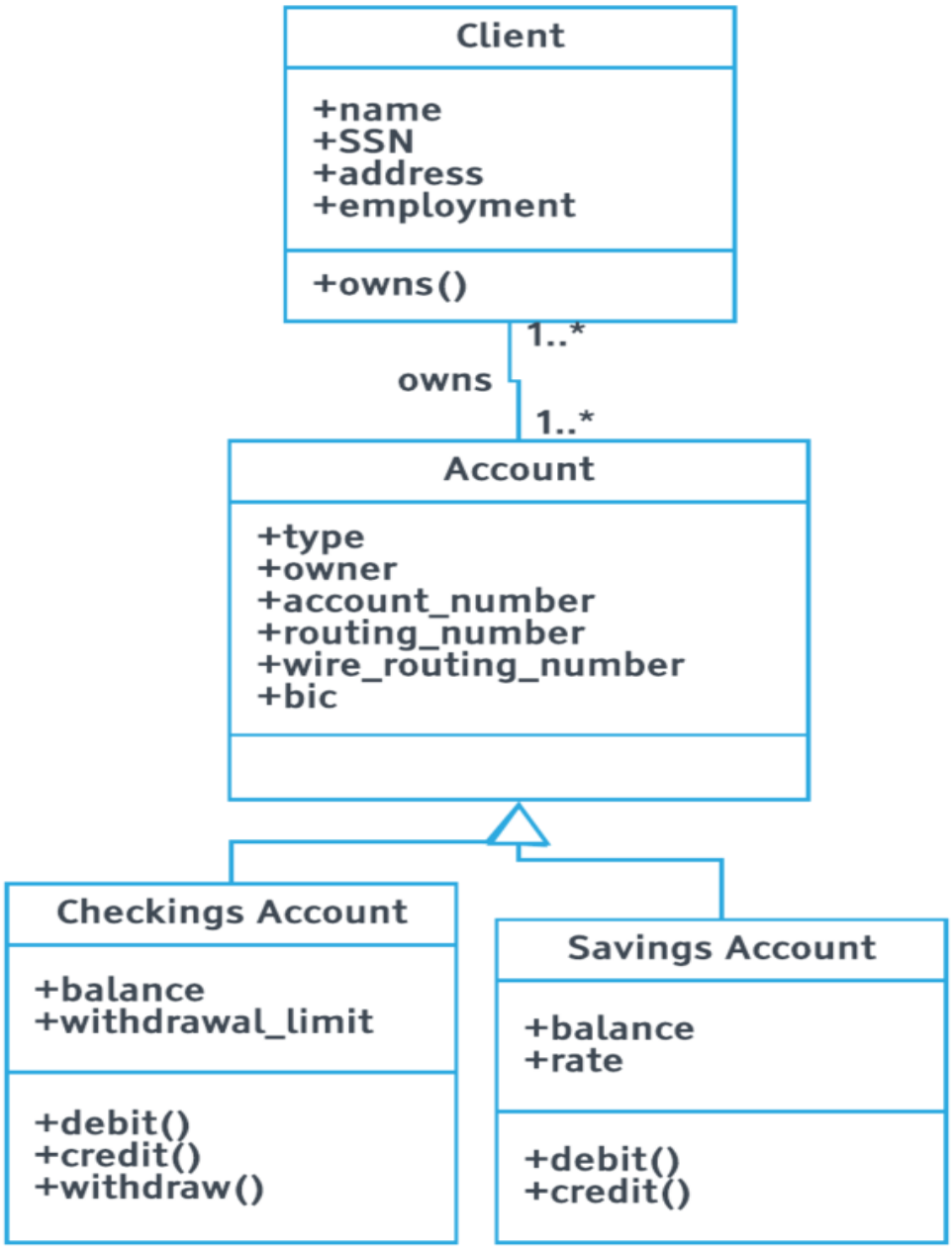
- Traverse the table and print all stored contacts.

Step 8: Stop

FLOW DIAGRAM



CLASS DIAGRAM OF THE PROBLEM STATEMENT



TOOLS AND TECHNOLOGIES

1. Programming Language:
   → C / C++ / Java / Python (used to implement Robin Hood Hashing and directory functions)
2. Data Structure:
   → Hash Table (with Robin Hood Hashing technique for efficient searching and insertion)
3. Database (optional):
   → MySQL / SQLite / MongoDB (for storing contact information permanently)
4. IDE / Editor:
   → Visual Studio Code / Code::Blocks / PyCharm (for development and testing)
5. Version Control:
   → Git and GitHub (for code management)
6. User Interface (optional):
   → HTML, CSS, JavaScript or simple Console-based UI (for interacting with users)
7. Operating System:
   → Windows / Linux (for running the system)
