DATA STRUCTURES AND ALGORITHMS

MOBILE PHONEBOOK APPLICATION PROJECT

GROUP PROJECT SUBMITTED BY:
Uetusuvera Tjivau 224077031

Ndapewa Shikongeni 224001256

Vekotora U Ngeama 221116303

Paulinus Angula 224053701

Lettycia Mateus 223068411

TABLE OF CONTENT

PROJECT OVERVIEW

This project aims to develop an efficient phonebook application for a Namibian telecommunications company. The application will utilize simple linear data structures (arrays and linked lists) to perform typical phonebook operations: inserting, searching, displaying, deleting, updating, and optionally sorting contacts.

In this this modern world an efficient contact management is extremely important for service providers.The ability to access and manage customer information quickly can improve customer service and build better communication between the company and its clients.

Purpose

The primary purpose of this phonebook application is to provide a straightforward interface for storing and managing contact information. This application will focus on basic operations, ensuring that users can quickly and easily insert, search, display, delete, and update contact information.

The application will include the following key functionalities:

1. **Insert Contact:** Users can add new contacts, ensuring that they can store essential customer information.
2. **Search Contact:** Users can search for contacts by name, allowing for quick access to important information.
3. **Display All Contacts:** A feature to display all stored contacts will enable users to view their entire contact list easily.
4. **Delete Contact:** Users can remove contacts from the phonebook, ensuring that the contact list remains up to date.
5. **Update Contact:** This functionality allows users to modify existing contact details, such as updating a phone number.
6. **Sort Contacts:** Users can sort contacts alphabetically, which can improve search efficiency and usability.
7. **Analyze Search Efficiency:** The application will include a mechanism to measure the efficiency of the search algorithm, providing insights into its performance.
8. **Exit**

Modules and Functions

1. **Data Structure Module**
    - **Contact Class:** Defines the structure for individual contacts, including attributes for name and phone number.
    - **Phonebook Class:** Manages the collection of contacts and implements core functionalities.
2. **Functionality Modules**
    - **Insert Contact**
        - **Function:** insert_contact(new_contact)
        - **Description:** Adds a new contact to the phonebook. Checks for capacity limits before insertion.
    - **Search Contact**
        - **Function:** search_contact(name)
        - **Description:** Searches for a contact by name and returns the contact details or a not-found message.
    - **Display All Contacts**
        - **Function:** display_contacts()
        - **Description:** Displays all contacts stored in the phonebook in a readable format.
    - **Delete Contact**
        - **Function:** delete_contact(name)
        - **Description:** Removes a contact from the phonebook by name and confirms deletion or returns a not-found message.

- **Update Contact**
  - **Function:** update_contact(name, new_phone_number)
  - **Description:** Updates the phone number of an existing contact and returns a success or not-found message.
- **Sort Contacts (Optional)**
  - **Function:** sort_contacts()
  - **Description:** Sorts contacts alphabetically by name for improved organization and faster searching.
- **Analyze Search Efficiency**
  - **Function:** analyze_search_efficiency(name)
  - **Description:** Measures and reports the time taken to search for a contact, providing insights into the search algorithm's performance.

JAVA CODE

```
import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;


// Class to represent a contact with a name and phone number

class Contact {

  String name;

  String phoneNumber;


  // Constructor to initialize a contact

  Contact(String name, String phoneNumber) {

    this.name = name;

    this.phoneNumber = phoneNumber;

  }

}


// Main class for the phonebook application

public class phone {

  private List<Contact> phonebook; // List to store contacts

  private Scanner scanner; // Scanner for user input


  // Constructor to initialize the phonebook and scanner

  public phone() {

    phonebook = new ArrayList<>();
```

```
        scanner = new Scanner(System.in);
    }

    // Main method to start the application
    public static void main (String[] args) {
        phone Scanner = new phone(); // Create an instance of the
phonebook
        Scanner.mainMenu(); // Display the main menu
    }

    // Method to display the main menu and handle user choices
    private void mainMenu() {
        while (true) {
            // Display menu options
            System.out.println("Phonebook Menu:");
            System.out.println("1. Insert Contact");
            System.out.println("2. Search Contact");
            System.out.println("3. Display All Contacts");
            System.out.println("4. Delete Contact");
            System.out.println("5. Update Contact");
            System.out.println("6. Sort Contacts");
            System.out.println("7. Analyze Search Efficiency");
            System.out.println("8. Exit");
            System.out.print("Choose an option: ");

            int option = scanner.nextInt(); // Read user choice
            scanner.nextLine();  // Consume newline
```

```
// Handle user choice using switch statement
switch (option) {
  case 1:
    insertContact(); // Insert a new contact
    break;
  case 2:
    searchContact(); // Search for a contact
    break;
  case 3:
    displayAllContacts(); // Display all contacts
    break;
  case 4:
    deleteContact(); // Delete a contact
    break;
  case 5:
    updateContact(); // Update a contact's phone number
    break;
  case 6:
    sortContacts(); // Sort contacts by name
    break;
  case 7:
    analyzeSearchEfficiency(); // Analyze search efficiency
    break;
  case 8:
    System.out.println("Exiting..."); // Exit the application
```

```
                return;
            default:
                System.out.println("Invalid option, please try again."); //
Handle invalid input
        }
      }
    }

    // Method to insert a new contact
    private void insertContact() {
        System.out.print("Enter name: ");
        String name = scanner.nextLine(); // Read name input
        if (name.isEmpty()) {
            System.out.println("Name cannot be empty."); // Check for empty
name
            return;
        }
        System.out.print("Enter phone number: ");
        String phoneNumber = scanner.nextLine(); // Read phone number
input
        phonebook.add(new Contact(name, phoneNumber)); // Add new
contact to the phonebook
        System.out.println("Contact added: " + name);
    }

    // Method to search for a contact by name
    private void searchContact() {
        System.out.print("Enter name to search: ");
        String name = scanner.nextLine(); // Read name to search
        Contact result = phonebook.stream() // Search for the contact
            .filter(contact -> contact.name.equalsIgnoreCase(name))
            .findFirst()
            .orElse(null);

        if (result != null) {
          // If contact is found, display the details
          System.out.println("Contact found: Name = " + result.name + ",
Phone Number = " + result.phoneNumber);
        } else {
          System.out.println("Contact not found."); // If not found, notify
the user
        }
      }

      // Method to display all contacts in the phonebook
      private void displayAllContacts() {
        if (phonebook.isEmpty()) {
          System.out.println("Phonebook is empty."); // Check if
phonebook is empty
        } else {
          // Iterate and display each contact
          for (Contact contact : phonebook) {
            System.out.println("Name: " + contact.name + ", Phone
Number: " + contact.phoneNumber);
```

```
        }
      }
    }

    // Method to delete a contact by name
    private void deleteContact() {
      System.out.print("Enter name to delete: ");
      String name = scanner.nextLine(); // Read name to delete
      Contact contactToRemove = searchContactByName(name); //
Search for the contact
      if (contactToRemove != null) {
        phonebook.remove(contactToRemove); // Remove the contact if
found
        System.out.println("Contact deleted: " + name);
      } else {
        System.out.println("Contact not found."); // Notify if not found
      }
    }

    // Method to update a contact's phone number
    private void updateContact() {
      System.out.print("Enter name to update: ");
      String name = scanner.nextLine(); // Read name to update
      Contact contactToUpdate = searchContactByName(name); //
Search for the contact
      if (contactToUpdate != null) {
        System.out.print("Enter new phone number: ");
```

```
          String newPhoneNumber = scanner.nextLine(); // Read new phone
number
          contactToUpdate.phoneNumber = newPhoneNumber; // Update
the phone number
          System.out.println("Contact updated: " + name);
        } else {
          System.out.println("Contact not found."); // Notify if not found
        }
    }

    // Method to sort contacts alphabetically by name
    private void sortContacts() {
        phonebook.sort(Comparator.comparing(contact ->
contact.name)); // Sort contacts
        System.out.println("Contacts sorted.");
    }

    // Method to analyze search efficiency
    private void analyzeSearchEfficiency() {
        System.out.println("Search Time Complexity: O(n)"); // Inform user
about time complexity
    }

    // Helper method to search for a contact by name
    private Contact searchContactByName(String name) {
        return phonebook.stream()
            .filter(contact -> contact.name.equalsIgnoreCase(name)) //
Search for contact
```

```
            .findFirst()
            .orElse(null);
    }
}
```

PSEUDOCODE

```
STRUCTURE Contact
  STRING name
  STRING phoneNumber
END STRUCTURE



  DECLARE phonebook AS List of Contacts



  FUNCTION Main
    INITIALIZE phonebook as an empty list

    WHILE (true) DO

     DISPLAY "Phonebook Menu:"
      DISPLAY "1. Insert Contact"
      DISPLAY "2. Search Contact"
      DISPLAY "3. Display All Contacts"
      DISPLAY "4. Delete Contact"
      DISPLAY "5. Update Contact"
      DISPLAY "6. Sort Contacts"
      DISPLAY "7. Analyze Search Efficiency"
      DISPLAY "8. Exit"
      DISPLAY "Choose an option:"
      GET option


      IF (option==1) THEN
         DISPLAY "Enter name:"
         GET name
         DISPLAY "Enter phone number:"
         GET phoneNumber
         CALL InsertContact(phonebook, name, phoneNumber)

      ELSE IF (option==2) THEN
         DISPLAY "Enter name to search:"
         GET name
         result = CALL SearchContact(phonebook, name)
         IF (result != null) THEN
            DISPLAY "Contact found: Name =", result.name, ",
Phone Number =", result.phoneNumber
         ELSE
            DISPLAY "Contact not found"
         END IF

      ELSE IF (option==3) THEN
```

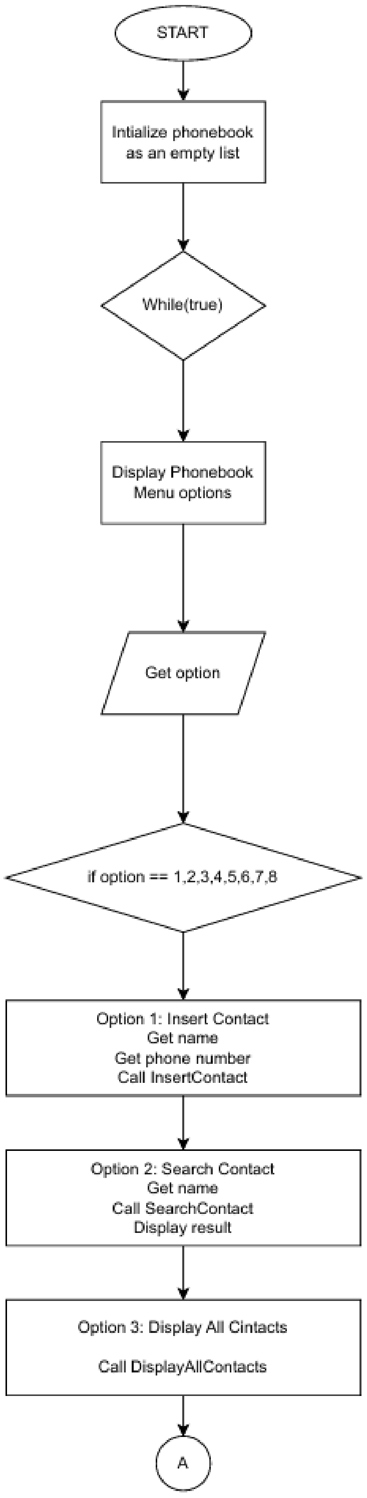
```
    CALL DisplayAllContacts(phonebook)

ELSE IF (option==4) THEN
    DISPLAY "Enter name to delete:"
    GET name
    CALL DeleteContact(phonebook, name)

ELSE IF (option==5) THEN
    DISPLAY "Enter name to update:"
    GET name
    DISPLAY "Enter new phone number:"
    GET newPhoneNumber
    CALL UpdateContact(phonebook, name,
newPhoneNumber)

ELSE IF (option==6) THEN
    CALL SortContacts(phonebook)

ELSE IF (option==7) THEN
    CALL AnalyzeSearchEfficiency()

ELSE IF (option==8) THEN
    DISPLAY "Exiting..."
```

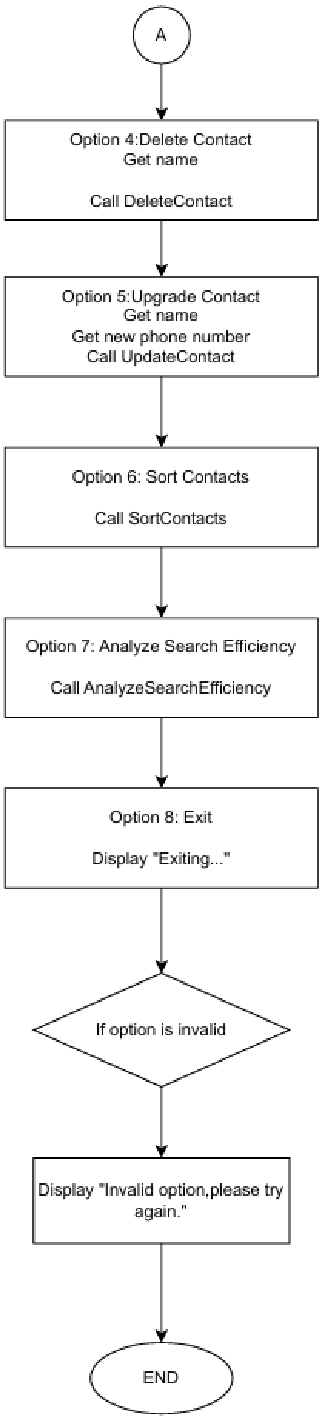
```
        ELSE
            DISPLAY "Invalid option, please try again."
        END IF
    END WHILE
END FUNCTION

FUNCTION InsertContact(phonebook, name, phoneNumber)
    DECLARE newContact AS Contact
    newContact.name = name
    newContact.phoneNumber = phoneNumber
    ADD newContact TO phonebook
    DISPLAY "Contact added:", name
END FUNCTION


FUNCTION SearchContact(phonebook, name) RETURNS
Contact
    FOR EACH contact IN phonebook DO
        IF (contact.name == name) THEN
            RETURN contact
        END IF
    END FOR
    RETURN null
END FUNCTION
```

```
FUNCTION DisplayAllContacts(phonebook)
 THEN
    DISPLAY "Phonebook is empty."
  ELSE
    FOR EACH contact IN phonebook DO
      DISPLAY "Name:", contact.name, ", Phone Number:",
contact.phoneNumber
    END FOR
  END IF
END FUNCTION


FUNCTION DeleteContact(phonebook, name)
  DECLARE contactToRemove AS Contact
  contactToRemove = CALL SearchContact(phonebook,
name)
  IF (contactToRemove != null) THEN
    REMOVE contactToRemove FROM phonebook
    DISPLAY "Contact deleted:", name
  ELSE
    DISPLAY "Contact not found."
  END IF
END FUNCTION



FUNCTION UpdateContact(phonebook, name,
newPhoneNumber)
  DECLARE contactToUpdate AS Contact
  contactToUpdate = CALL SearchContact(phonebook,
name)
  IF (contactToUpdate != null) THEN
    contactToUpdate.phoneNumber = newPhoneNumber
    DISPLAY "Contact updated:", name
  ELSE
    DISPLAY "Contact not found."
  END IF
END FUNCTION



FUNCTION SortContacts(phonebook)
  SORT(phonebook, contact.name)
  DISPLAY "Contacts sorted."
END FUNCTION



FUNCTION AnalyzeSearchEfficiency()
```

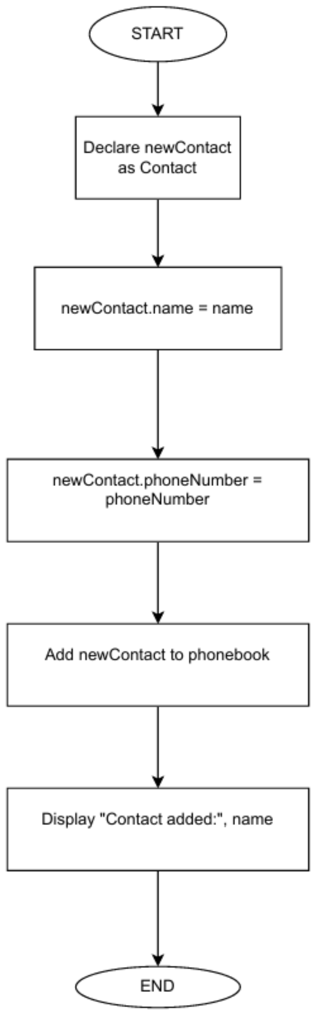
```
    DISPLAY "Search Time Complexity: O(n)"
END FUNCTION
```

MAIN FLOWCHART

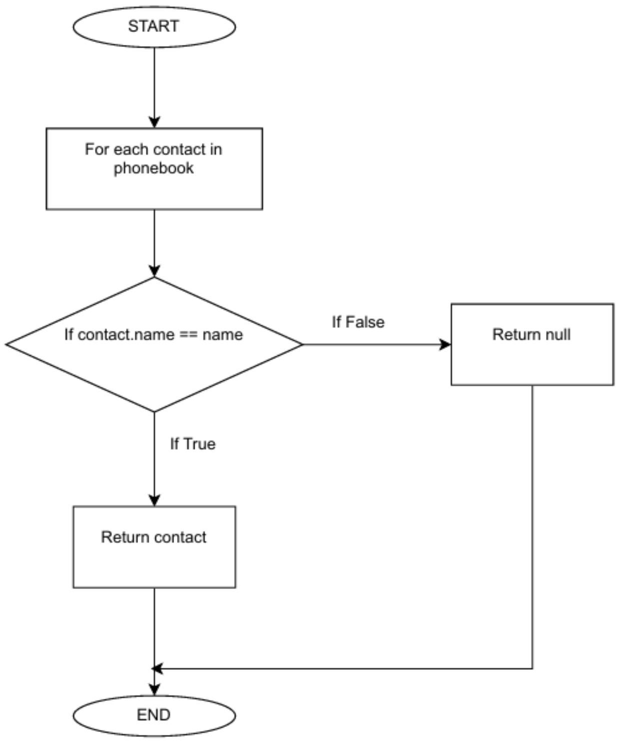



A
Option 4:Delete Contact
Get name
Call DeleteContact
Option 5:Upgrade Contact
Get name
Get new phone number
Call UpdateContact
Option 6: Sort Contacts
Call SortContacts
Option 7: Analyze Search Efficiency
Call AnalyzeSearchEfficiency
Option 8: Exit
Display "Exiting..."
If option is invalid
Display "Invalid option,please try again."
END

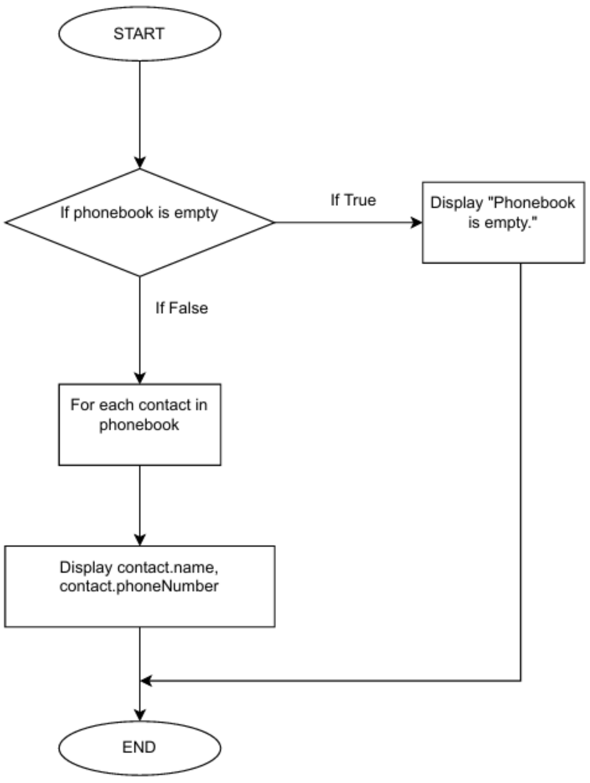
1. InsertContact Function Flowchart


START
Declare newContact as Contact
newContact.name = name
newContact.phoneNumber = phoneNumber
Add newContact to phonebook
Display "Contact added:", name
END

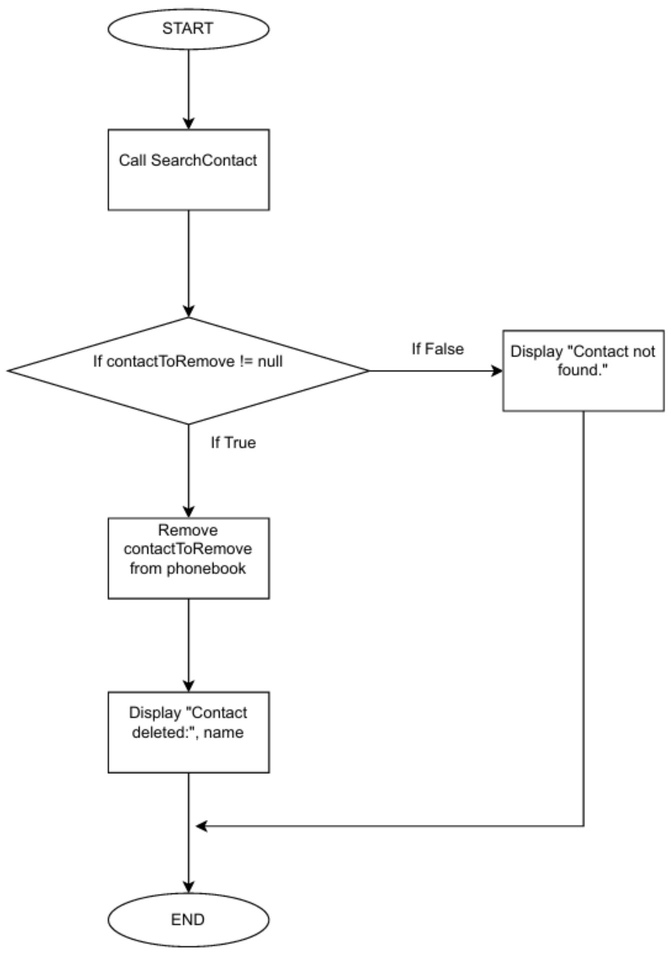
2. SearchContact Function Flowchart



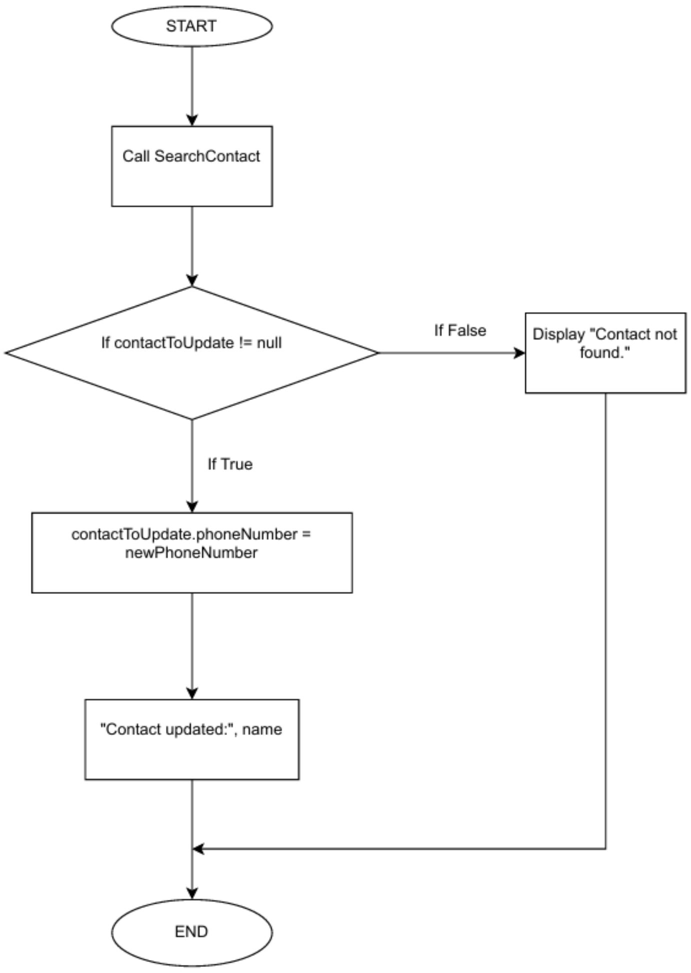
3. DisplayAllContacts Function Flowchart


START
If phonebook is empty
If True
Display "Phonebook is empty."
If False
For each contact in phonebook
Display contact.name, contact.phoneNumber
END

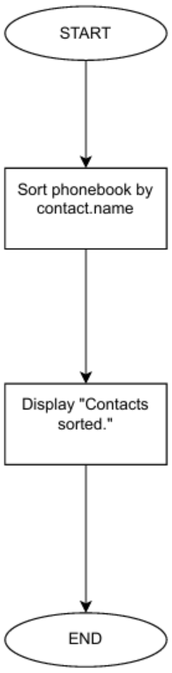
4. DeleteContact Function Flowchart


START
Call SearchContact
If contactToRemove != null
If False
Display "Contact not found."
If True
Remove contactToRemove from phonebook
Display "Contact deleted:", name
END

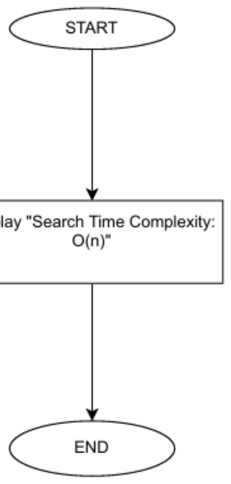
5. UpdateContact Function Flowchart


START
Call SearchContact
If contactToUpdate != null
If False
Display "Contact not found."
If True
contactToUpdate.phoneNumber = newPhoneNumber
"Contact updated:", name
END

6. SortContacts Function Flowchart



7. AnalyzeSearchEfficiency Function Flowchart



CONTRIBUTIONS:

Uetusuvera Tjivau 224077031 AND Ndapewa Shikongeni 224001256 - LEAD DEVELOPERS: RESPONSIBLE FOR IMPLEMENTING THE CORE FUNCTIONALITIES OF THE APPLICATION,AND DESIGNED THE USER INTERFACE

Vekotora U Ngeama 221116303 AND Lettycia Mateus 223068411 - DOCUMENTATION SPECIALISTS: CREATED A DETAILED DOCUMENTATION FOR THE APPLICATION

Paulinus Angula 224053701: ASSISTED WITH USER INTERFACE DESIGN.