Data Structure and Algorithm

Topic: Namibia Telecommunication Company Phonebook

Group Members:

1. Dumisani Ngwira 220104689
2. Deon Kayele 221090142
3. Helvi Herman 223090999
4. James Ruben 224081918
5. Thomas Kufuna 218109881
6. Christiano April 224064398

Section A: Pseudocode

```
1. Insert Contact
Start
   Input: Contact Name, Contact Number
   If the phonebook is not full:
      Insert the contact at the next available position
      Display "Contact added successfully."
   Else:
      Display "Phonebook is full."
End
2. Search Contact
Start
   Input: Search Query (Contact Name or Number)
   For each contact in the phonebook:
      If the contact matches the query:
         Display the contact details
         Exit
   If no match is found:
      Display "Contact not found."
End
3. Display All Contacts
Start
   If phonebook is not empty:
      For each contact in the phonebook:
         Display contact details
   Else:
      Display "Phonebook is empty."
End
4. Delete Contact
Start
   Input: Contact Name or Number
   If the contact exists:
```

```
        Remove the contact from the phonebook
        Shift all remaining contacts up
        Display "Contact deleted successfully."
    Else:
        Display "Contact not found."
End
5. Update Contact
Start
    Input: Contact Name or Number
    If the contact exists:
        Input: New Contact Name or Number
        Update the contact information
        Display "Contact updated successfully."
    Else:
        Display "Contact not found."
End
6. Sort Contacts (Optional)
Start
    Sort the phonebook contacts in alphabetical order by name
    Display "Contacts sorted."
End
```

7. Efficiency Analysis (Search Algorithm)

Linear Search: Time Complexity is O(n), where n is the number of contacts.

Section B: JAVA

```
import java.util.ArrayList;
import java.util.Scanner;
class Contact {
    String name;
    String phoneNumber;
    public Contact(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }
    @Override
    public String toString() {
        return "Name: " + name + ", Phone Number: " + phoneNumber;
    }
}
public class PhoneBook {
    private ArrayList<Contact> contacts;
    public PhoneBook() {
```

```
        contacts = new ArrayList<>();
    }
    // Insert Contact
    public void insertContact(String name, String phoneNumber) {
        contacts.add(new Contact(name, phoneNumber));
        System.out.println("Contact added successfully.");
    }
    // Search Contact
    public void searchContact(String query) {
        for (Contact contact : contacts) {
            if (contact.name.equalsIgnoreCase(query) ||
contact.phoneNumber.equals(query)) {
                System.out.println("Contact found: " + contact);
                return;
            }
        }
        System.out.println("Contact not found.");
    }
    // Display All Contacts
    public void displayContacts() {
        if (contacts.isEmpty()) {
            System.out.println("Phonebook is empty.");
        } else {
            for (Contact contact : contacts) {
                System.out.println(contact);
            }
        }
    }
    // Delete Contact
    public void deleteContact(String query) {
        for (int i = 0; i < contacts.size(); i++) {
            if (contacts.get(i).name.equalsIgnoreCase(query) ||
contacts.get(i).phoneNumber.equals(query)) {
                contacts.remove(i);
                System.out.println("Contact deleted successfully.");
                return;
            }
        }
        System.out.println("Contact not found.");
    }
    // Update Contact
    public void updateContact(String query, String newName, String newPhoneNumber)
{
        for (Contact contact : contacts) {
```

```
        if (contact.name.equalsIgnoreCase(query) ||
contact.phoneNumber.equals(query)) {
            contact.name = newName;
            contact.phoneNumber = newPhoneNumber;
            System.out.println("Contact updated successfully.");
            return;
        }
    }
    System.out.println("Contact not found.");
  }
  public static void main(String[] args) {
    PhoneBook phoneBook = new PhoneBook();
    Scanner scanner = new Scanner(System.in);
    while (true) {
      System.out.println("\nPhoneBook 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. Exit");
      System.out.print("Choose an option: ");
      int choice = scanner.nextInt();
      scanner.nextLine(); // Consume newline
      switch (choice) {
        case 1:
          System.out.print("Enter Name: ");
          String name = scanner.nextLine();
          System.out.print("Enter Phone Number: ");
          String phoneNumber = scanner.nextLine();
          phoneBook.insertContact(name, phoneNumber);
          break;
        case 2:
          System.out.print("Enter Name or Phone Number to search: ");
          String query = scanner.nextLine();
          phoneBook.searchContact(query);
break;
        case 3:
          phoneBook.displayContacts();
break;
        case 4:
          System.out.print("Enter Name or Phone Number to delete: ");
          query = scanner.nextLine();
          phoneBook.deleteContact(query);
```

```
break;
            case 5:
               System.out.print("Enter Name or Phone Number to update: ");
               query = scanner.nextLine();
               System.out.print("Enter New Name: ");
               String newName = scanner.nextLine();
               System.out.print("Enter New Phone Number: ");
               String newPhoneNumber = scanner.nextLine();
               phoneBook.updateContact(query, newName, newPhoneNumber);
               break;
            case 6:
               System.out.println("Exiting...");
System.exit(0);
            default:
               System.out.println("Invalid choice. Try again.");
         }
      }
   }
}
```

Flowchart