## SECTION A

```
Module Phonebook in pseudocode
    Data Structure:
        Contact
            String name
            String phoneNumber
            Contact next

        Phonebook
            Contact head

    Function InitializePhonebook()
        head = NULL

    Function InsertContact(name: String,
phoneNumber: String)
        newContact = new Contact
        newContact.name = name
        newContact.phoneNumber =
phoneNumber
        newContact.next = head
        head = newContact

    Function SearchContact(name: String) ->
Contact
        current = head
        while current is not NULL
            if current.name == name
                return current
            end if
            current = current.next
        end while
        return NULL

    Function DisplayAllContacts()
        current = head
        while current is not NULL
            Print current.name + ": " +
current.phoneNumber
```

```
        current = current.next

    Function DeleteContact(name: String) ->
Boolean
      current = head
      previous = NULL
      while current is not NULL
        if current.name == name
          if previous is not NULL
            previous.next = current.next
          else
            head = current.next
          end if
          return TRUE
        end if
        previous = current
        current = current.next
      end while
      return FALSE

    Function UpdateContact(name: String,
newPhoneNumber: String) -> Boolean
        contact = SearchContact(name)
        if contact is not NULL
            contact.phoneNumber =
newPhoneNumber
            return TRUE
        else
            return FALSE

    Function SortContacts()
        if head is NULL or head.next is NULL
            return

        // Convert linked list to array for
sorting
        contactsArray = []
        current = head
        while current is not NULL
            contactsArray.append(current)
            current = current.next
    // Sort array by name (simple bubble sort for demonstration)
    for i from 0 to length(contactsArray) - 1
        for j from 0 to length(contactsArray) - i - 1
            if contactsArray[j].name > contactsArray[j + 1].name
                // Swap
                temp = contactsArray[j]
                contactsArray[j] = contactsArray[j + 1]
                contactsArray[j + 1] = temp
            end if
        end for
    end for

    // Convert sorted array back to linked list
    head = contactsArray[0]
    current = head
    for i from 1 to length(contactsArray) - 1
```

```
            current.next = contactsArray[i]
            current = current.next
        end for
        current.next = NULL

    Function AnalyzeSearchEfficiency()
        // This function can calculate the
number of comparisons made during a
search
        // For simplicity, we can return a fixed
value representing the worst-case scenario
        maxComparisons = 0
        current = head
        while current is not NULL
            maxComparisons += 1
            current = current.next
        end while
        return maxComparisons
End Module
```

## SECTION  B

Phonebook Implementation in Java

```
class Contact {
    String name;
    Int phoneNumber;
    Contact next;

    public Contact(String name, Int 
phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.next = null;
    }
}

class Phonebook {
    private Contact head;

    public Phonebook() {
        head = null;
    }

    public void insertContact(String name,
String phoneNumber) {
        Contact newContact = new
Contact(name, phoneNumber);
        newContact.next = head;
        head = newContact;
    }

    public Contact searchContact(String
name) {
        Contact temp = head;
        while (temp != null) {
            if (temp.name.equals(name)) {
                return temp;
            }
            temp = temp.next;
        }
        return null;
    }

    public void displayAllContacts() {
        Contact temp = head;
        if (temp == null) {
            System.out.println("Phonebook is
empty.");
            return;
        }
        while (temp != null) {
            System.out.println(temp.name + ": "
+ current.phoneNumber);
            temp = temp.next;
        }
    }

    public boolean deleteContact(String
name) {
        Contact temp = head;
        Contact previous = null;
        while (temp != null) {
            if (temp.name.equals(name)) {
                if (previous != null) {
                    previous.next = temp.next;
                } else {
                    head = temp.next;
                }
                return true;
            }
            previous = temp;
            temp = temp.next;
        }
        return false;
    }

    public boolean updateContact(String
name, Int newPhoneNumber) {
        Contact contact =
searchContact(name);
```

```
        if (contact != null) {
            contact.phoneNumber =
newPhoneNumber;
            return true;
        } else {
            return false;
        }
    }

    public void sortContacts() {
        if (head == null || head.next == null) {
            return;
        }

        // Convert linked list to array for
sorting
        java.util.ArrayList<Contact>
contactsList = new java.util.ArrayList<>();
        Contact temp = head;
        while (temp != null) {
            contactsList.add(temp);
            temp = temp.next;
        }

        // Sort array by INSERTION
        for (int i = 0; i < contactsList.size(); i+
+) {
            for (int j = 0; j < contactsList.size() - i
- 1; j++) {
                if
(contactsList.get(j).name.compareTo(cont
actsList.get(j + 1).name) > 0) {
                    // Swap
                    Contact temp =
contactsList.get(j);
                    contactsList.set(j,
contactsList.get(j + 1));
                    contactsList.set(j + 1, temp);
                }
            }
        }
        // Convert sorted array back to linked
list
        head = contactsList.get(0);
        temp = head;
        for (int i = 1; i < contactsList.size(); i+
+) {
            temp.next = contactsList.get(i);
            temp = temp.next;
        }
        temp.next = null;
    }

    public int analyzeSearchEfficiency() {
        int maxComparisons = 0;
        Contact temp = head;
        while (temp != null) {
            maxComparisons++;
            temp = temp.next;
        }
        return maxComparisons;
    }
    public static void main(String[] args) {
        Phonebook phonebook = new
Phonebook();

        // Insert contacts
        phonebook.insertContact("Joseph",
"081-456-7890");
        phonebook.insertContact("Dudu",
"081-765-4321");
        phonebook.insertContact("moyo",
"081-555-5555");

        // Display all contacts
        System.out.println("All Contacts:");
        phonebook.displayAllContacts();

        // Search for a contact
        Contact contact =
phonebook.searchContact("joseph");
        if (contact != null) {
            System.out.println("Found: " +
contact.name + " - " +
contact.phoneNumber);
        } else {
            System.out.println("Contact not
found.");
        }

        // Update a contact
        if
(phonebook.updateContact("matamu",
"081-222-3333")) {
            System.out.println("matamu's
contact updated.");
        } else {
            System.out.println("Contact not
found to update.");
        }

        // Delete a contact
        if
```

```
(phonebook.deleteContact("James")) {
            System.out.println("James deleted
from phonebook.");
        } else {
            System.out.println("Contact not
found to delete.");
        }

        // Display all contacts after deletion
        System.out.println("Contacts after
deletion:");
        phonebook.displayAllContacts();

        // Sort contacts
        phonebook.sortContacts();
        System.out.println("Contacts after
sorting:");
        phonebook.displayAllContacts();

        // Analyze search efficiency
        int efficiency =
phonebook.analyzeSearchEfficiency();
        System.out.println("Maximum
comparisons for search: " + efficiency);
    }
}
```