**A black background with a black square

Description automatically generated with medium confidence**

[Group Project 2024](https://elearning.nust.na/mod/assign/view.php?id=65376)

**Data Structures and Algorithms 1\_S2\_2024**

|  |  |  |
| --- | --- | --- |
| **SN** | **Name** | **Student Number** |
| 1 | Blasius K.Pinias **(Team Leader)** | 218117345 |
| 2 | Paulus Frans | 221119922 |
| 3 | Cherley Hambira | 224095099 |
| 4 | Kavindika Ngondeka | 224093444 |

# Section A: Documentation for Phonebook Application

## Project Overview

The aim of this project is to develop a mobile phonebook application for a Namibian telecommunications company. The application will utilize basic linear data structures, such as arrays, to implement essential phonebook operations. The key functionalities include adding, searching, deleting, updating, and displaying contacts. Additionally, an optional feature for sorting contacts will be implemented to enhance search efficiency. This documentation provides a detailed description of the algorithms used for each operation.

## Functional Requirements

The phonebook application will support the following operations:

1. Insert Contact
2. Search Contact
3. Display All Contacts
4. Delete Contact
5. Update Contact
6. Sort Contacts (optional)
7. Analyze the Efficiency of the Search Algorithm

## Algorithms Representation

### 1. Insert Contact

**Description:** This function is responsible for adding a new contact to the phonebook. It takes the contact's name and phone number as inputs, creates a new contact object, and appends it to the existing phonebook array. After the contact is successfully added, it returns a confirmation message.

### 2. Search Contact

**Description:** This function searches for a contact within the phonebook using either the contact's name or phone number as a search term. It iterates through the phonebook array to find a match. If a contact is found, it returns the contact details; if not, it returns a message indicating that the contact was not found.

### 3. Display All Contacts

**Description:** This function displays all the contacts currently stored in the phonebook. If the phonebook is empty, it returns a message stating that no contacts are available. Otherwise, it iterates through the phonebook and prints the name and phone number of each contact.

### 4. Delete Contact

**Description:** This function allows the user to delete a contact from the phonebook by specifying either the contact's name or phone number. It searches through the phonebook and removes the contact from the array if a match is found. Upon successful deletion, it returns a confirmation message; if the contact is not found, it returns an appropriate message.

### 5. Update Contact

**Description:** This function enables the user to update the details of an existing contact. By providing the name or phone number of the contact to be updated, along with the new name and phone number, the function searches for the contact in the phonebook and updates its information. It returns a success message if the update is performed; otherwise, it notifies the user that the contact was not found.

### 6. Sort Contacts (Optional)

**Description:** This optional function sorts the contacts in the phonebook based on their names. This feature can enhance the efficiency of searching for contacts, making it easier for users to locate them. Once the sorting is completed, it returns a confirmation message.

### 7. Analyze the Efficiency of the Search Algorithm

**Description:** This function analyzes and reports the efficiency of the search algorithm used in the application. It evaluates the performance in terms of time complexity, providing insights on the best and worst-case scenarios. The analysis helps users understand the efficiency of contact retrieval from the phonebook.

## Conclusion

This documentation outlines the design and purpose of each function in the phonebook application based on simple linear data structures. Each function is modular and addresses specific phonebook operations, ensuring clarity and ease of implementation. The optional sorting function can be utilized to enhance search performance, while the efficiency analysis provides insights into the algorithm's performance.