Assignment 1: Part 1

Title: Phone Book Application

This assignment focuses on the development of a Phone Book application, which serves as a digital address book for efficiently managing contacts. The assignment comprises three main components: a brief outline of the application, an explanation of the data structure and algorithms using Flowcharts and Pseudocode, and the formulation of a test plan.

Outline of the application

The Phone Book application serves as a digital record, enabling users to store and manage their contacts efficiently. The objective of the application is to provide a user-friendly interface for performing various operations on contacts, such as adding, deleting, searching, viewing, and sorting contacts, as well as searching for specific contacts. By utilizing a command-line interface, users can easily input commands to interact with the Phone Book and perform the desired actions on their contacts. This makes the application intuitive and accessible for users to effectively manage their contact information.

To execute the application:

- Launch the Phone Book program.
- Follow the on-screen instructions and input the corresponding number for the desired action.

- Perform the desired action, such as adding, viewing, deleting, or searching for contacts.
- Repeat the process until the user chooses to exit the application.

Design of data structure and algorithms

The Phone Book application utilizes a dictionary, also known as an associative array, to store the contacts. The contacts are stored as key-value pairs, where the contact name serves as the key and the phone number as the value. The following operations are supported:

1. Displaying: The application includes an operation to display the existing contact list. If the Phone Book is empty, it returns an error message.
2. Insertion: When adding a new contact, the application checks if the contact already exists. If not, the contact name and phone number are added as a key-value pair to the Phone Book. If the contact exists, it returns an error message.
3. Deletion: When deleting a contact, the application first checks if the contact exists. If found, it is removed from the Phone Book using the key. If not found, it returns an error message.
4. Sorting: The contacts in the Phone Book are initially stored in the order of insertion. When the user requests to sort the contacts by contact name as the key, the application utilizes the bubble sort algorithm to rearrange the contacts in alphabetical order based on the contact names.
5. Searching: To search for a specific contact, the application checks if the contact exists in the Phone Book using the contact name as the key. If found, the

corresponding contact name and phone number are displayed. If not found, it returns an error message.

6. Existing: All operations above return to the main menu after completion until the user chooses to exit.

Please refer to **Appendix I** for the Flowcharts that illustrating the operation of the Phone Book application, and **Appendix II** for the Pseudocode demonstrating the algorithms used in the application.

Test Plan

To ensure the functionality and correctness of the Phone Book application, the following test plan is recommended:

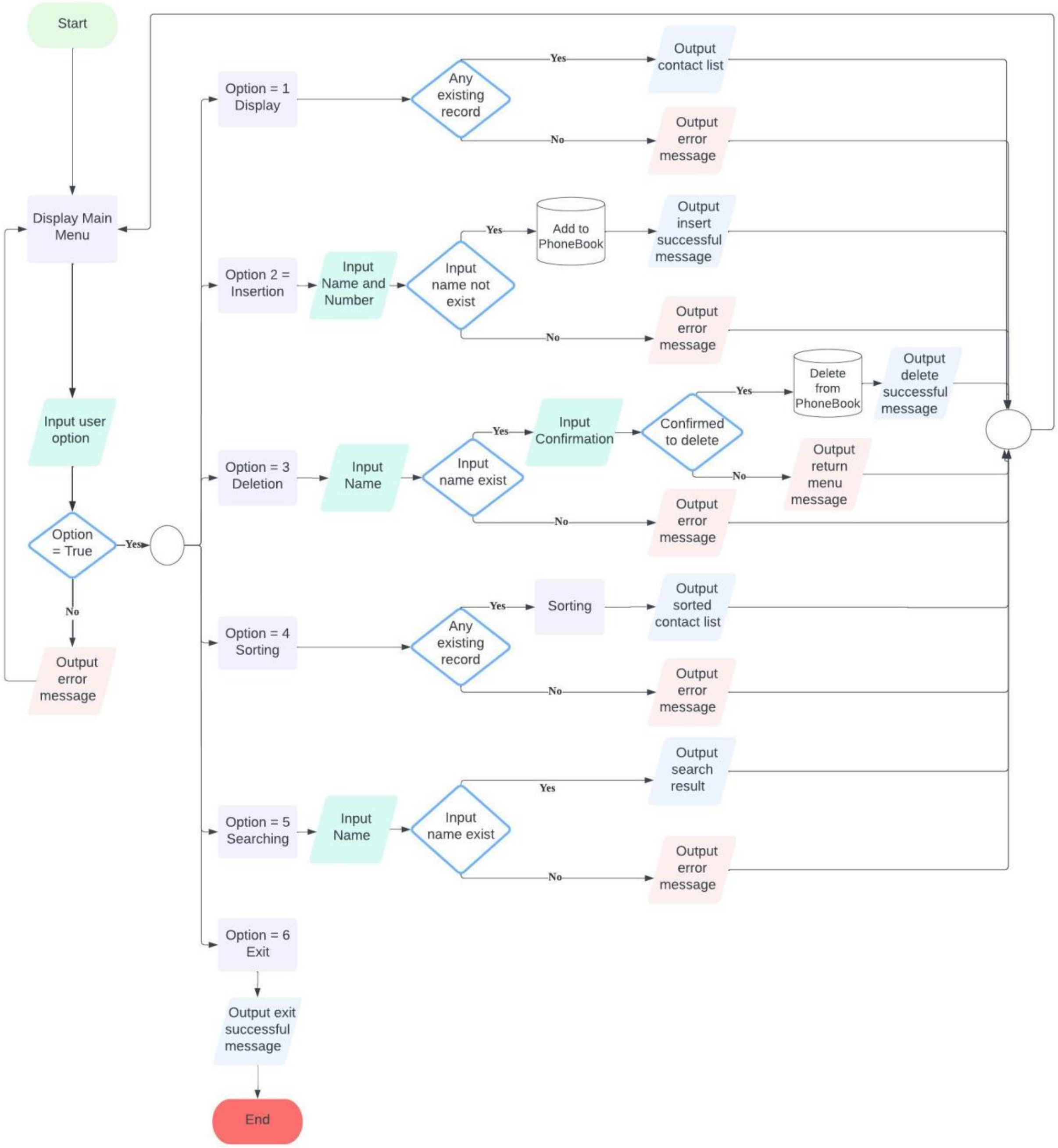
1. Test name: Display existing contacts and adding a new contact.
    - Input “1” to view the current contacts.
    - Input “2” to add a new contact.
    - Enter three to five phone number and contact name.
    - Input “1” to view the updated list of contacts and verify that the contact is successfully saved.
2. Test name: Deleting a contact.
    - Input “3” to delete a contact.
    - Enter the name of the contact to be deleted.
    - Confirm the deletion.
    - Input “1” to view the updated list of contacts and check if the contact is removed.

3. Test name: Sorting contacts.
    - Input “4” to sort the contacts.
    - Verify that the contacts are displayed in alphabetical order based on the contact names.
4. Test name: Searching for a contact.
    - Input “5” to search for a contact.
    - Enter the name of the contact to be searched.
    - Verify if the contact exists and if the correct phone number is displayed.
5. Test name: Invalid input handling.
    - Enter invalid inputs at various stages, such as entering alphabetic characters instead of numbers or providing non-existent contact names.
    - Verify if the program displays appropriate error messages and return to the main menu.
6. Test name: Exist the application.
    - Input “6” to verify if the program successfully exits.

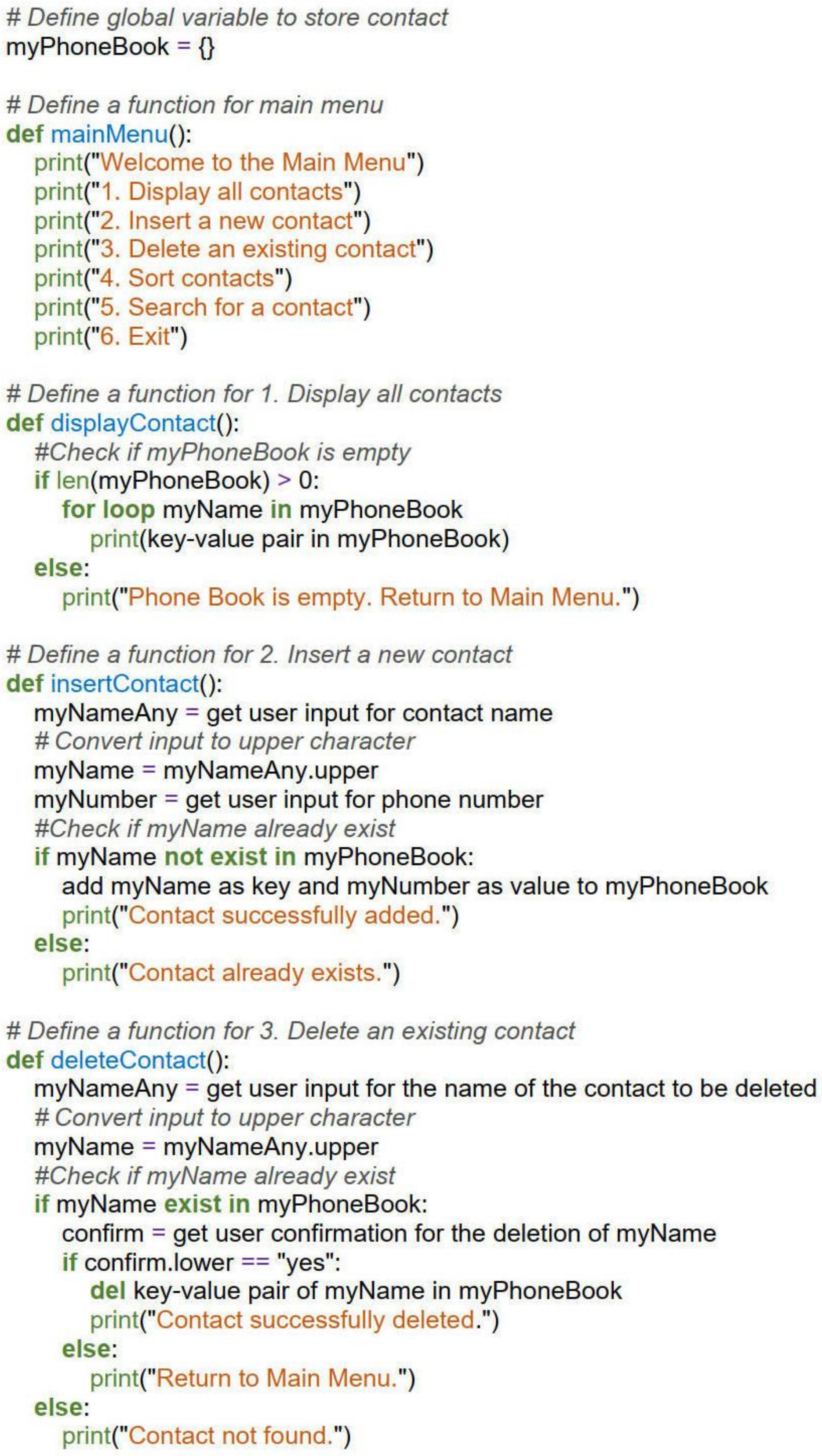
The test plan covers the major functionalities of the application. By conducting these tests, we can ensure that the Phone Book application functions as intended, handles user inputs correctly, and provides accurate results.

In conclusion, the Phone Book application provides users with an efficient and user-friendly command-line interface to digitally manage their contacts. By adhering to the provided test plan and ensuring the accurate implementation of algorithms and data structures, the Phone Book application can become a reliable tool for users to organize their contact information effectively.

Appendix I   Flowcharts of the Phone Book application



Appendix II Pseudocode of the Phone Book application



```
# Define global variable to store contact
myPhoneBook = {}

# Define a function for main menu
def mainMenu():
    print("Welcome to the Main Menu")
    print("1. Display all contacts")
    print("2. Insert a new contact")
    print("3. Delete an existing contact")
    print("4. Sort contacts")
    print("5. Search for a contact")
    print("6. Exit")

# Define a function for 1. Display all contacts
def displayContact():
    #Check if myPhoneBook is empty
    if len(myPhoneBook) > 0:
        for loop myName in myPhoneBook
            print(key-value pair in myPhoneBook)
    else:
        print("Phone Book is empty. Return to Main Menu.")

# Define a function for 2. Insert a new contact
def insertContact():
    myNameAny = get user input for contact name
    # Convert input to upper character
    myName = myNameAny.upper
    myNumber = get user input for phone number
    #Check if myName already exist
    if myName not exist in myPhoneBook:
        add myName as key and myNumber as value to myPhoneBook
        print("Contact successfully added.")
    else:
        print("Contact already exists.")

# Define a function for 3. Delete an existing contact
def deleteContact():
    myNameAny = get user input for the name of the contact to be deleted
    # Convert input to upper character
    myName = myNameAny.upper
    #Check if myName already exist
    if myName exist in myPhoneBook:
        confirm = get user confirmation for the deletion of myName
        if confirm.lower == "yes":
            del key-value pair of myName in myPhoneBook
            print("Contact successfully deleted.")
        else:
            print("Return to Main Menu.")
    else:
        print("Contact not found.")
```

```
# Define a function for 4. Sort contacts
def sortContact():
    sortedContacts = list of keys in myPhoneBook
    #Check if myPhoneBook is empty
    if len(myPhoneBook) > 0:
        # Implement bubble sort
        n = len(sortedContacts)
        for i in range(n - 1):
            for j in range(n - 1 - i):
                if sortedContacts[j] > sortedContacts[j + 1]:
                    # Swap the elements
                    Swap sortedContacts[j] with sortedContacts[j + 1]
        print("Contacts sorted by alphabetical order:")
        for loop myName in sortedContacts:
            print(key-value pair in sortedContacts)
    else:
        print("Phone Book is empty. Return to Main Menu.")

# Define a function for 5. Search for a contact
def searchContact():
    myNameAny = get user input for the name to search
    # Convert input to upper character
    myName = myNameAny.upper
    #Check if myName already exist
    if myName exist in myPhoneBook:
        print(key-value pair in myPhone Book)
    else:
        print("Contact not found.")

# Define Phone Book Application
def myPhoneBookApp():
    while loop when myOption is True:
        mainMenu()
        myOption = get user input as integer
        if myOption == "1":
            displayContact()
        elif myOption == "2":
            insertContact()
        elif myOption == "3":
            deleteContact()
        elif myOption == "4":
            sortContact()
        elif myOption == "5":
            searchContact()
        elif myOption == "6":
            print("Exiting the application.")
            break
        else:
            print("Invalid input, please try again.")

# Run program
myPhoneBookApp()
```