**A black background with a black square

Description automatically generated with medium confidence**

**DSA521S DATA STRUCTURES AND ALGORITHMS, SEMESTER 2, 2024**

**GROUP ASSIGNMENT PRESENTATION: SCORESHEET AND PROJECT DESCRIPTION TEMPLATE**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SN** | **Name** | **Student Number** | **Specialisation** | **Role played in the project** | **Mark over 100% (for lecturers use only)** |
| 1 | Kiami Quinga | 224008714 | Informatics | Pseudocode |  |
| 2 | Joeseph Katamba | 224046764 | Informatics | Flowchart |  |
| 3 | Morag McLaughlan | 224081640 | Cyber Security | Modules, Functions and analysis of search algorithm |  |
| 4 | Hope Shetekela | 224014439 | Computer Science | Modules, Functions and analysis of search algorithm |  |
| 5 | Takunda Vhere | 224041576T | Computer Science | Flowchart |  |
| 6 | Simataa Ntelamo | 224079859 | Cyber Security | Source code |  |

**Name of Team Leader (TL):**

Kiami Quinga

**TITLE OF PROJECT:**

Phonebook

**DATE:**

13/10/2024

# Modules:

* Insert Contact
* Search Contact
* Display All Contacts
* Delete Contact
* Update contact
* Sort Contacts

## Functions:

Insert Contact:

Purpose: Adds a new contact to the phonebook.

Description: This function creates a new contact entry and add it to the phonebook list.

Search Contact:

Purpose: Searches for a contact by either name or phone.

Description: Performs a linear search through the phonebook to find a matching contact based on the query.

Display All Contacts:

Purpose: Outputs all contacts stored in the phonebook.

Description: This function checks if the phonebook is empty and, if not, iterates through and prints each contact's details.

Delete Contact:

Purpose: Removes a specified contact from the phonebook.

Description: This function searches for a contact based on a query, then deletes it if found.

Update Contact:

Purpose: Updates the details of an existing contact.

Description: The function finds a contact by query, then updates its name and phone details with the provided new information.

Sort Contacts:

Purpose: Sorts contacts alphabetically by name.

Description: This optional function uses a bubble sort algorithm to arrange contacts in alphabetical order, improving search efficiency.

## Analyse the efficiency of your search algorithm:

If you have an unsorted list, searching requires checking each item one by one, which takes O(n) time. Sorting the list first allows for quicker searches with binary search, but sorting itself takes O(n log n) time. Sorting only makes sense if you plan to search the list multiple times, as it incurs an initial time cost.

## Pseudocode:

Insert Contact:

Function InsertContact(name, phone):

*// Create a new contact with the details provided by the user*

contact = CreateContact(name, phone)

*// Add the new contact to the phonebook*

Add contact to phonebook

*// Confirm the contact has been added*

Return "Contact added successfully"

Search Contact:

Function SearchContact(query):

*// Loop through each contact in the phonebook*

For each contact in phonebook:

*// Check if the contact name or phone number matches the query*

If contact.name == query OR contact.phone == query:

// Return the contact details if found

Return contact

*// Notify that the contact was not found*

Return "Contact not found"

Display All Contacts:

Function DisplayAllContacts():

*// Check if the phonebook is empty*

If phonebook is empty:

*// Notify that there are no contacts available*

Print "No contacts available"

Else:

*// Loop through and display each contact's details*

For each contact in phonebook:

Print contact.name, contact.phone

Delete Contact:

Function DeleteContact(query):

*// Loop through each contact in the phonebook*

For each contact in phonebook:

*// Check if contact name or phone number matches the query*

If contact.name == query OR contact.phone == query:

*// Remove the matched contact from the phonebook*

Remove contact from phonebook

*// Confirm the contact has been deleted*

Return "Contact deleted"

*// Notify that the contact was not found*

Return "Contact not found"

Update contact:

Function UpdateContact(query, new\_name, new\_phone):

*// Loop through each contact in the phonebook*

For each contact in phonebook:

*// Check if contact name or phone number matches the query*

If contact.name == query OR contact.phone == query:

*// Update the contact's name and phone number with new details*

contact.name = new\_name

contact.phone = new\_phone

*// Confirm the contact has been updated*

Return "Contact updated"

*// Notify that the contact was not found*

Return "Contact not found"

Sort Contacts:

Function SortContacts():

*// Loop through the phonebook for sorting*

For i = 0 to length(phonebook) - 1:

*// Nested loop for comparing and swapping contacts*

For j = 0 to length(phonebook) - i - 1:

*// Check if current contact is greater than the next contact*

If phonebook[j].name > phonebook[j+1].name:

*// Swap the two contacts*

Swap phonebook[j] and phonebook[j+1]

## Combined Pseudocode:

*// Define the phonebook as an empty list*

phonebook = []

*// Function to insert a new contact into the phonebook*

Function InsertContact(name, phone):

contact = CreateContact(name, phone) *// Create a new contact with the provided details*

Add contact to phonebook *// Add the contact to the phonebook*

Return "Contact added successfully"

*// Function to search for a contact by name or phone number*

Function SearchContact(query):

For each contact in phonebook: *// Loop through each contact in the phonebook*

If contact.name == query OR contact.phone == query:

Return contact *// Return the contact if a match is found*

Return "Contact not found" *// If no match is found, return a message*

*// Function to display all contacts in the phonebook*

Function DisplayAllContacts():

If phonebook is empty:

Print "No contacts available" *// Notify if there are no contacts*

Else:

For each contact in phonebook:

Print contact.name, contact.phone *// Print the name and phone number of each contact*

*// Function to delete a contact by name or phone number*

Function DeleteContact(query):

For each contact in phonebook:

If contact.name == query OR contact.phone == query:

Remove contact from phonebook *// Remove the contact if a match is found*

Return "Contact deleted" *// Confirm the deletion*

Return "Contact not found" *// If no match is found, return a message*

*// Function to update an existing contact's details*

Function UpdateContact(query, new\_name, new\_phone):

For each contact in phonebook:

If contact.name == query OR contact.phone == query:

contact.name = new\_name *// Update the contact's name*

contact.phone = new\_phone *// Update the contact's phone number*

Return "Contact updated" *// Confirm the update*

Return "Contact not found" *// If no match is found, return a message*

*// Optional Function to sort contacts alphabetically by name*

Function SortContacts():

For i = 0 to length(phonebook) - 1:

For j = 0 to length(phonebook) - i - 1:

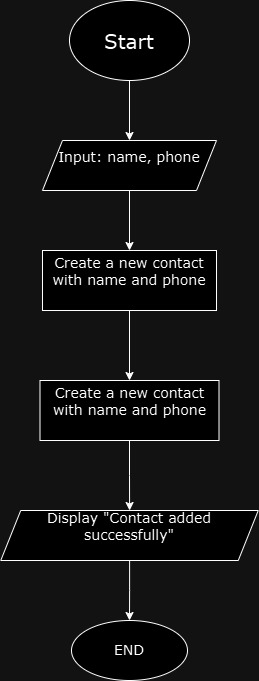
If phonebook[j].name > phonebook[j+1].name: *// Check if the current contact is greater than the next one*

Swap phonebook[j] and phonebook[j+1] *// Swap contacts to sort alphabetically*

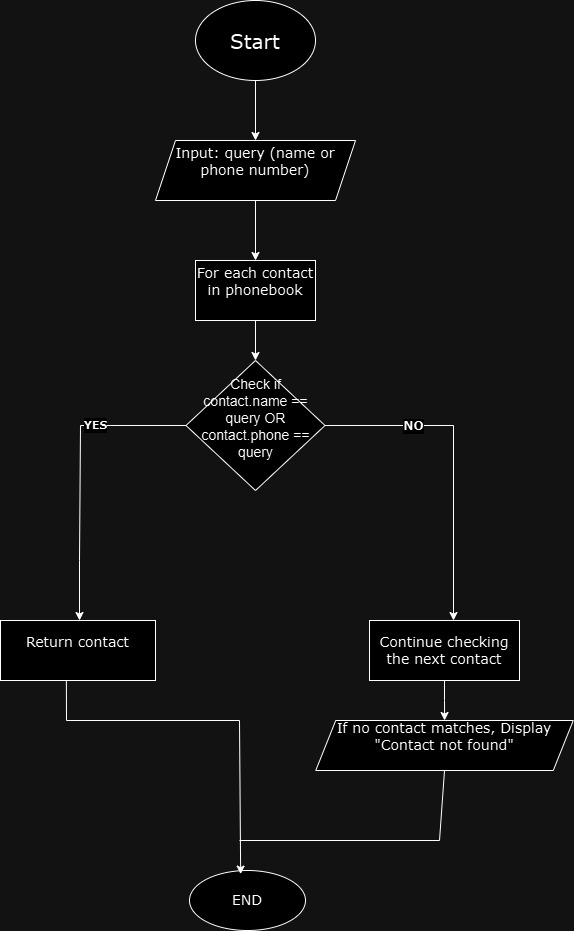
Return "Contacts sorted" *// Confirm that sorting is complete*

# Flowcharts:

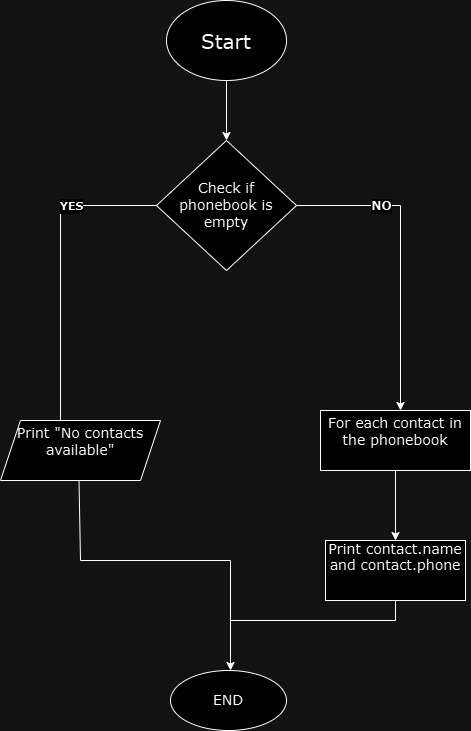
Insert Contact:



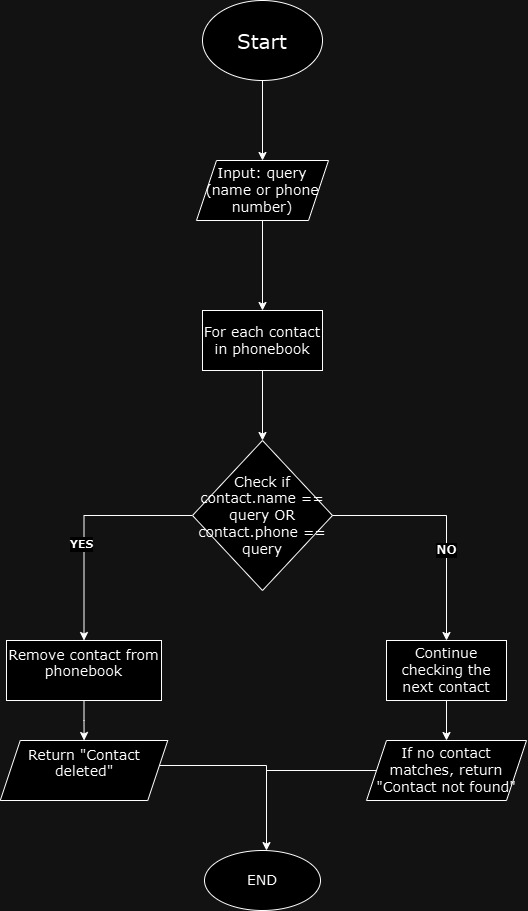
Search Contact:



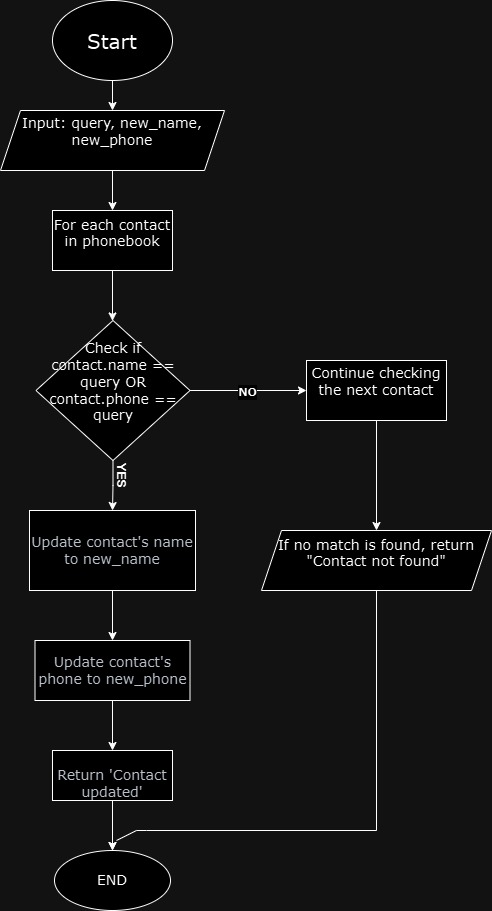
Display Contacts:



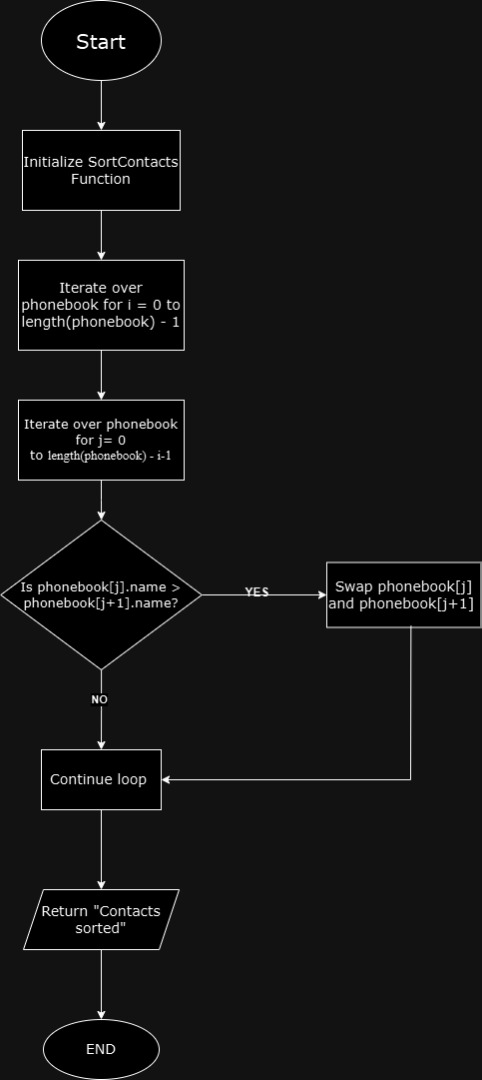
Delete Contact:



Update Contact:



Sort Contacts:



# Section B:

Algorithm:

class Phonebook:

def \_\_init\_\_(self):

"""Initialize the phonebook as an empty list."""

self.phonebook = []

def create\_contact(self, name, phone):

"""Create a contact dictionary with name and phone."""

return {"name": name, "phone": phone}

def insert\_contact(self, name, phone):

"""Insert a new contact into the phonebook."""

contact = self.create\_contact(name, phone) # Create a new contact

self.phonebook.append(contact) # Add the contact to the phonebook

return "Contact added successfully" # Return success message

def search\_contact(self, query):

"""Search for a contact by name or phone number."""

for contact in self.phonebook: # Loop through each contact in the phonebook

if contact["name"] == query or contact["phone"] == query:

return contact # Return the contact if found

return "Contact not found" # If no match is found, return a message

def display\_all\_contacts(self):

"""Display all contacts in the phonebook."""

if not self.phonebook: # Check if the phonebook is empty

print("No contacts available") # Notify if there are no contacts

else:

for contact in self.phonebook: # Loop through and display each contact's details

print(f"Name: {contact['name']}, Phone: {contact['phone']}")

def delete\_contact(self, query):

"""Delete a contact by name or phone number."""

for contact in self.phonebook: # Loop through each contact in the phonebook

if contact["name"] == query or contact["phone"] == query:

self.phonebook.remove(contact) # Remove the contact if found

return "Contact deleted" # Confirm deletion

return "Contact not found" # If no match is found, return a message

def update\_contact(self, query, new\_name, new\_phone):

"""Update the details of an existing contact."""

for contact in self.phonebook: # Loop through each contact in the phonebook

if contact["name"] == query or contact["phone"] == query:

contact["name"] = new\_name # Update the contact's name

contact["phone"] = new\_phone # Update the contact's phone number

return "Contact updated" # Confirm update

return "Contact not found" # If no match is found, return a message

def sort\_contacts(self):

"""Sort the contacts in the phonebook by name."""

self.phonebook.sort(key=lambda contact: contact["name"]) # Sort the phonebook list

return "Contacts sorted" # Confirm sorting

def main(self):

"""Main function to display the phonebook menu and handle user input."""

while True:

# Display the phonebook menu options

print("\nPhonebook Menu:")

print("1. Insert Contact")

print("2. Search Contact")

print("3. Display All Contacts")

print("4. Delete Contact")

print("5. Update Contact")

print("6. Sort Contacts")

print("7. Exit")

choice = input("Enter your choice: ") # Get user input for the menu choice

# Call the appropriate function based on user input

if choice == "1":

name = input("Enter name: ") # Get name input

phone = input("Enter phone: ") # Get phone input

print(self.insert\_contact(name, phone)) # Insert contact and print result

elif choice == "2":

query = input("Enter name or phone to search: ") # Get search query

print(self.search\_contact(query)) # Search contact and print result

elif choice == "3":

self.display\_all\_contacts() # Display all contacts

elif choice == "4":

query = input("Enter name or phone to delete: ") # Get query for deletion

print(self.delete\_contact(query)) # Delete contact and print result

elif choice == "5":

query = input("Enter name or phone to update: ") # Get query for update

new\_name = input("Enter new name: ") # Get new name input

new\_phone = input("Enter new phone: ") # Get new phone input

print(self.update\_contact(query, new\_name, new\_phone)) # Update contact and print result

elif choice == "6":

print(self.sort\_contacts()) # Sort contacts and print result

elif choice == "7":

print("Exiting Phonebook...") # Exit message

break # Exit the loop and end the program

else:

print("Invalid choice, please try again.") # Notify if the choice is invalid

# Entry point of the program

if \_\_name\_\_ == "\_\_main\_\_":

phonebook\_app = Phonebook() # Create an instance of the Phonebook class

phonebook\_app.main() # Call the main function to run the application