DATA STRUCTURE AND ALGORITHM PROJECT

GROUP 13

Section A .1: Algorithm representation Design for Phonebook Application.

INTRODUCTION

This algorithm is designed to manage a simple phonebook system using basic data structures. The goal is to create an efficient and user-friendly way to handle common phonebook tasks, such as adding, searching for, displaying, deleting, and updating contacts.

TWO MODULES INCLUDED:

1.CONTACT MANAGEMENT MODULE

*Insert contacts
* search contacts
* delete contacts
*display all contacts
*update contacts
*Sort contacts
*Analyze Efficiency

2.NOTIFICATION AND ALERT MODULE

*Display Success Messages:
*Show Error Messages
*Confirmation Dialogs
*Warning Alerts
*Information Messages

LOGIC REPRESENTATION

1.1 INSERT CONTACTS
-Adds a new contact to the phonebook, ensuring that no duplicate entries are created and that the phonebook has space for new contacts.

1.2 SEARCH CONTACTS
-Allows users to find a contact by their name and retrieve their phone number.

1.3 DELETE CONTACTS
-Removes a contact from the phonebook based on their name.

1.4 DISPLAY ALL CONTACTS
-Shows a list of all contacts currently in the phonebook.

1.5 UPDATE CONTACTS
-Modifies the phone number of an existing contact.

1.6 SORT CONTACTS

-Arranges the contacts in alphabetical order by name to make searching faster.

1.7 ANALYZE EFFICIENCY
-Measures how long it takes to search for a contact, helping to understand the performance of the search operation.

2.1 DISPLAY SUCCESS MESSAGE
-Inform users when an operation, such as adding or updating a contact, is successful.

2.2 SHOW ERROR MESSAGES
-Notify users of any errors that occur during operations.

2.3 CONFIRM DIALOGS
-Ask for user confirmation before performing actions that modify or delete data.

2.4 WARNING ALERTS
-Alert users to potential issues or important information that requires attention.

2.5 INFORMATION MESSAGES
- Provide additional information or guidance to users.

1. CONTACT MANAGEMENT MODULE

1.1 INSERT CONTACTS FUNCTION

INPUTS	PROCESS	OUTPUTS
• `phonebook` (`name` and `phone_number`) • `new_contact` (`name` and `phone_number`) • `size` (maximum number of contacts the phonebook can hold)	Check if the Phonebook is Full Check for Duplicate Contacts Find an Empty Slot and Insert the New Contact	"Contact added successfully" or "Phonebook is full" or "Contact already exists").

PSEUDOCODE:

```
START
Algorithm InsertContact(phonebook, new_contact, size):
    Input: phonebook (array of Contact), new_contact (Contact), size (int)
    Output: Message (string)

    // Step 1: Check if the phonebook is full
    If length(phonebook) >= size:
        Print "Phonebook is full"
        Return

    // Step 2: Check for duplicate contacts
```

```
For i = 0 to length(phonebook) - 1:
    If phonebook[i] is not empty and phonebook[i].name == new_contact.name:
        Print "Contact already exists"
        Return

// Step 3: Find an empty slot and insert the new contact
For i = 0 to length(phonebook) - 1:
    If phonebook[i] is empty:
        phonebook[i] = new_contact
        Print "Contact added successfully"
        Return

// If no empty slot is found
Print "No available slot for new contact"
  END!!
```

1.2 SEARCH CONTACTS FUNCTION

INPUTS	PROCESS	OUTPUTS
• `phonebook` (Array of Contact objects, where each Contact has `name` and `phone_number`) • `search_name` (String)	• Loop through each contact in the phonebook. • If a contact with the name matching `search_name` is found, return the phone number. • If the loop finishes without finding a match, return "Contact not found."	The phone number of the contact if found, or a "Contact not found" message.

PSEUDOCODE:

```
Algorithm SearchContact(phonebook, search_name):
    Input: phonebook (array of Contact), search_name (string)
    Output: Phone number or "Contact not found" (string)

    For i = 0 to length(phonebook) - 1:
        If phonebook[i] is not empty and phonebook[i].name == search_name:
            Return phonebook[i].phone_number

    Return "Contact not found"
```

1.3 DELETE CONTACTS FUNCTION

INPUTS	PROCESS	OUTPUTS
• `phonebook` (Array of Contact objects) • `contact_name` (String)	• Check if the phonebook is empty. • If it is not empty, loop through each contact and print their details. • If it is empty, print "No contacts available.	A message indicating whether the contact was successfully deleted or not found.

PSEUDOCODE:

```
Algorithm DeleteContact(phonebook, contact_name):
    Input: phonebook (array of Contact), contact_name (string)
    Output: Message (string)

    For i = 0 to length(phonebook) - 1:
        If phonebook[i] is not empty and phonebook[i].name == contact_name:
            phonebook[i] = empty
            Print "Contact deleted successfully"
            Return

    Print "Contact not found"
```

1.4 DISPLAY ALL CONTACTS FUNCTION

INPUTS	PROCESS	OTPUTS
`phonebook` (Array of Contact objects)	1. Check if the phonebook is empty. 2. If it is not empty, loop through each contact and print their details. 3. If it is empty, print "No contacts available."	A list of contact details or a "No contacts available" message.

PSEUDOCODE:

```
Algorithm DisplayAllContacts(phonebook):
    Input: phonebook (array of Contact)
    Output: List of contact details or "No contacts available" (string)

    If length(phonebook) == 0:
        Print "No contacts available"
        Return
```

```
For i = 0 to length(phonebook) - 1:
    If phonebook[i] is not empty:
        Print phonebook[i].name, phonebook[i].phone_number
```

1.5 UPDATE CONTACTS FUNCTION

INPUTS	PROCESS	OUTPUTS
• `phonebook` (Array of Contact objects) • `contact_name` (String) • `new_phone_number` (String)	• Loop through each contact to find a match by `contact_name`. • If a match is found, update their phone number and indicate success. • If the loop finishes without finding the contact, return "Contact not found."	A message indicating whether the contact was successfully updated or not found.

PSEUDOCODE:

```
Algorithm UpdateContact(phonebook, contact_name, new_phone_number):
    Input: phonebook (array of Contact), contact_name (string), new_phone_number (string)
    Output: Message (string)

    For i = 0 to length(phonebook) - 1:
        If phonebook[i] is not empty and phonebook[i].name == contact_name:
            phonebook[i].phone_number = new_phone_number
            Print "Contact updated successfully"
            Return

    Print "Contact not found"
```

1.6 SORT CONTACTS FUNCTION

INPUTS	PROCESS	OUTPUTS
`phonebook` (Array of Contact objects)	• Use a simple sorting algorithm (e.g., Bubble Sort) to sort the contacts by name. • Print "Contacts sorted successfully."	The sorted phonebook.

PSEUDOCODE:

```
START
```

```
Algorithm SortContacts(phonebook):
   Input: phonebook (array of Contact)
   Output: Sorted phonebook

   For i = 0 to length(phonebook) - 2:
      For j = 0 to length(phonebook) - i - 2:
         If phonebook[j].name > phonebook[j + 1].name:
            Swap phonebook[j] with phonebook[j + 1]

   Print "Contacts sorted successfully"
END!!
```

1.7 ANALYZE EFFICIENCY FUNCTION

INPUTS	PROCESS	OUTPUTS
• `phonebook` (Array of Contact objects) • `search_name` (String)	• Start a timer before performing the search. • Call the `SearchContact` function. • Stop the timer and calculate the elapsed time. • Print the time taken for the search.	The time taken to perform the search.

PSEUDOCODE:

```
START
Algorithm AnalyzeSearchEfficiency(phonebook, search_name):
   Input: phonebook (array of Contact), search_name (string)
   Output: Time taken (float)

   Start timer
   Call SearchContact(phonebook, search_name)
   Stop timer
   elapsed_time = Get elapsed time from timer
   Print "Time taken to search: ", elapsed_time
END
```

2.NOTIFICATION AND ALERT MODULE

2.1 DISPLAY SUCCESS MESSAGES FUNCTION

INPUTS	PROCESS	OUTPUTS
`operation` (String indicating which operation was successful, e.g., "add" or "update")	• Check the operation type. • Print a success message	A message indicating the operation was successful.

	based on the operation.	

PSEUDOCODE:

START
Algorithm DisplaySuccessMessage(operation):
  Input: operation (string)
  Output: Success message (string)

  If operation == "add":
    Print "Success: Contact added successfully."
  Else If operation == "update":
    Print "Success: Contact updated successfully."
  Else:
    Print "Success: Operation completed successfully.
END!!

2.2 SHOW ERROR MESSAGES FUNCTION

INPUTS	PROCESS	OUTPUTS
`error_type` (String indicating the type of error, e.g., "full", "duplicate", "not_found")	• Check the type of error. • Print an error message based on the error type.	An error message indicating what went wrong.

PSEUDOCODE:

START
Algorithm ShowErrorMessage(error_type):
  Input: error_type (string)
  Output: Error message (string)

  If error_type == "full":
    Print "Error: Phonebook is full. Cannot add new contact."
  Else If error_type == "duplicate":
    Print "Error: Contact already exists."
  Else If error_type == "not_found":
    Print "Error: Contact not found."
  Else:
    Print "Error: An unknown error occurred.
END!!

2.3 CONFIRM DIALOGS FUNCTION

INPUTS	PROCESS	OUTPUTS
`action` (String indicating the action to be confirmed, e.g., "delete" or "modify")	• Ask the user if they want to proceed with the action.	User’s confirmation (Yes/No)

	• Return user’s response.	

PSEUDOCODE:

```
START
Algorithm ConfirmAction(action):
   Input: action (string)
   Output: User confirmation (string)

   Print "Are you sure you want to ", action, "? (Yes/No)"
   user_response = Get user input

   If user_response == "Yes":
      Return "Confirmed"
   Else:
      Return "Cancelled"
END!!
```

2.4 WARNING ALERTS FUNCTION

INPUTS	PROCESS	OUTPUTS
`warning_type` (String indicating the type of warning, e.g., "low_space", "unsaved_changes")	• Check the type of warning. • Print a warning message based on the warning type.	A warning message indicating the issue.

PSEUDOCODE:

```
START
Algorithm ShowWarningAlert(warning_type):
   Input: warning_type (string)
   Output: Warning message (string)

   If warning_type == "low_space":
      Print "Warning: The phonebook is nearly full."
   Else If warning_type == "unsaved_changes":
      Print "Warning: You have unsaved changes."
   Else:
      Print "Warning: Please check the system for potential issues.
END!!
```

2.5 INFORMATION MESSAGES FUNCTION

INPUTS	PROCESS	OUTPUTS
`info_type` (String indicating the type of information, e.g., "instructions", "format")	• Check the type of information needed. • Print a message	An information message providing guidance.

	providing the required information.	

PSEUDOCODE:

START

Algorithm ShowInformationMessage(info_type):
  Input: info_type (string)
  Output: Information message (string)

  If info_type == "instructions":
    Print "Information: To add a contact, enter the name and phone number."
  Else If info_type == "format":
    Print "Information: Phone numbers should be in the format (123) 456-7890."
END

  Else:
    Print "Information: Please refer to the user guide for more details."

Conclusion

This document has outlined the essential algorithms and pseudocode for managing our basic phonebook application, encompassing functionalities such as adding, updating, searching, and deleting contacts , notification and alert.