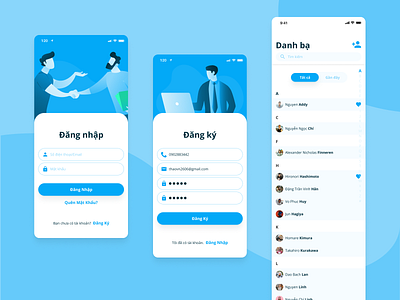
****

**Database Structure And Algorithm Project**

**PHONEBOOK APPLICATION**



|  |  |  |
| --- | --- | --- |
| NAME | STUDENT NUMBER | SPECIALISATION |
| Elifas Kahanda(**TL**) | 223043397 | Computer Science |
| Nakale Nelly | 223044148 | Informatics |
| Hailume David | 223096709 | Computer Science |
| Nghipunya Jennef | 223076228 | Informatics |
| Kakololo Saima | 223091405 | Computer Science |
| Mungandjela Jason | 222090537 | Computer Science |

**Data Structure**

We use a linked list to store contact information. Each node in the linked list is representing a contact and it contain the following fields:

* **Name** (string)
* **Phone Number** (string)
* **Next** (pointer to the next contact)

**Explanation Of Operations**

1.**Insert Contact**

* Create a new node
* Insert it at the end of the linked list

2. **Search Contact**

* Traverse the linked list from the head
* Compare each node’s name with the search term until a match is found or the end of the list is reached

3. **Display All Contacts**

* Traverse the linked list and print each contact’s details

4. **Delete Contact**

* Traverse the linked list to find the contact
* Adjust pointers to remove the node from the list

5. **Update Contact**

* Search for the contact
* Update its details if found

6. **Sort Contacts**

* Implementation using a simple sort algorithm (selection sort)

7. **Efficiency Analysis of Search Algorithm**

* The search operation has a time complexity of O(n)O(n) since it may require traversing all nodes in the worst case.

**Main Method**

The main method demonstrates how to use the phonebook class by performing various operations such as inserting contacts, searching, updating, deleting, and sorting.

**Efficiency Analysis**

* **Insert Operation:** O(1) for adding to the end of the linked list.
* **Search Operation:** O(n) in the worst case, as it requires a linear search through the list.
* **Display Operation:** O(n), as it involves iterating through the entire list.
* **Delete Operation:** O(n) in the worst case, as it requires searching for the contact before deletion.
* **Update Operation:** O(n) in the worst case, due to the search operation.
* **Sort Operation:** the selection sort algorithm has a time complexity of (O(n^2)).

This documentation outlines the design and implementation of a basic phonebook application using simple linear data structures. The application is designed to be easy to understand and implement, making it suitable for a telecommunications company’s needs.

**The Pseudocode**

START

**Phonebook = {}**

Print (“1. Add contact”)

Print (“2. Search contact”)

Print (“3. Display contact”)

Print (“4. Delete contact”)

Print (“5. Update contact”)

**AddContact Function** (name, phone number):

//Add the contact to the phone book

Phonebook [name]= phone number

Print “Contact added successfully”

**SearchContact Function** (name):

If name is in phonebook:

Print “name: “name, phone number:” phonebook(name)

Else

Print “Contact not found”

**DisplayContact Function ();**

If (phonebook == 0)

Print “Phonebook is empty”

Else

For each contact in phonebook:

Print (“Name:”, contact,”phone number:”, phonebook(contact)

**DeleteContact Function** (name):

If (name is in phonebook:)

Delete phonebook(name)

Print “Contact deleted successfully

Else

Print “Contact not found”

**UpdateContact Funtion** (name, newphone number):

contact = searchContact(name)

If (contact != null:)

contact.phone number = newphone number

Return true

else

Return false

**selectionSort Function** (array):

FOR (i = 0; i < n - 1; i++) //passes

min = i //set i to smallest element

FOR (j = i + 1; j < n; j++) //linear search for smallest element

IF (array[j] < array[min]) THEN //compare

min = j

ENDIF

ENDFOR

temp = array[i] //beginning of swapping

array[i] = array[min]

array[min] = temp //end of swapping

**Main Function**:

Initialize an empty list named phonebook

While true:

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"

Print "Enter your choice: "

Read userChoice

If (userChoice == 1:)

Print "Enter name: "

Read name

Print "Enter phone: "

Read phone

Call insertContact (name, phone)

Else if (userChoice == 2:)

Print "Enter name to search: "

Read name

contact = searchContact(name)

If (contact is not null:)

Print "Contact found: ", contact.name, contact.phone

Else:

Print "Contact not found"

Else if (userChoice == 3:)

Call displayContacts ()

Else if (userChoice == 4:)

Print "Enter name to delete: "

Read name

If (deleteContact(name):)

Print "Contact deleted"

Else:

Print "Contact not found"

Else if (userChoice == 5:)

Print "Enter name to update: "

Read name

Print "Enter new phone: "

Read newphone number

If (updateContact(name, newPhone number):)

Print "Contact updated"

Else:

Print "Contact not found"

Else if (userChoice == 6:)

Call sortContacts ()

Print "Contacts sorted"

Else if (userChoice == 7:)

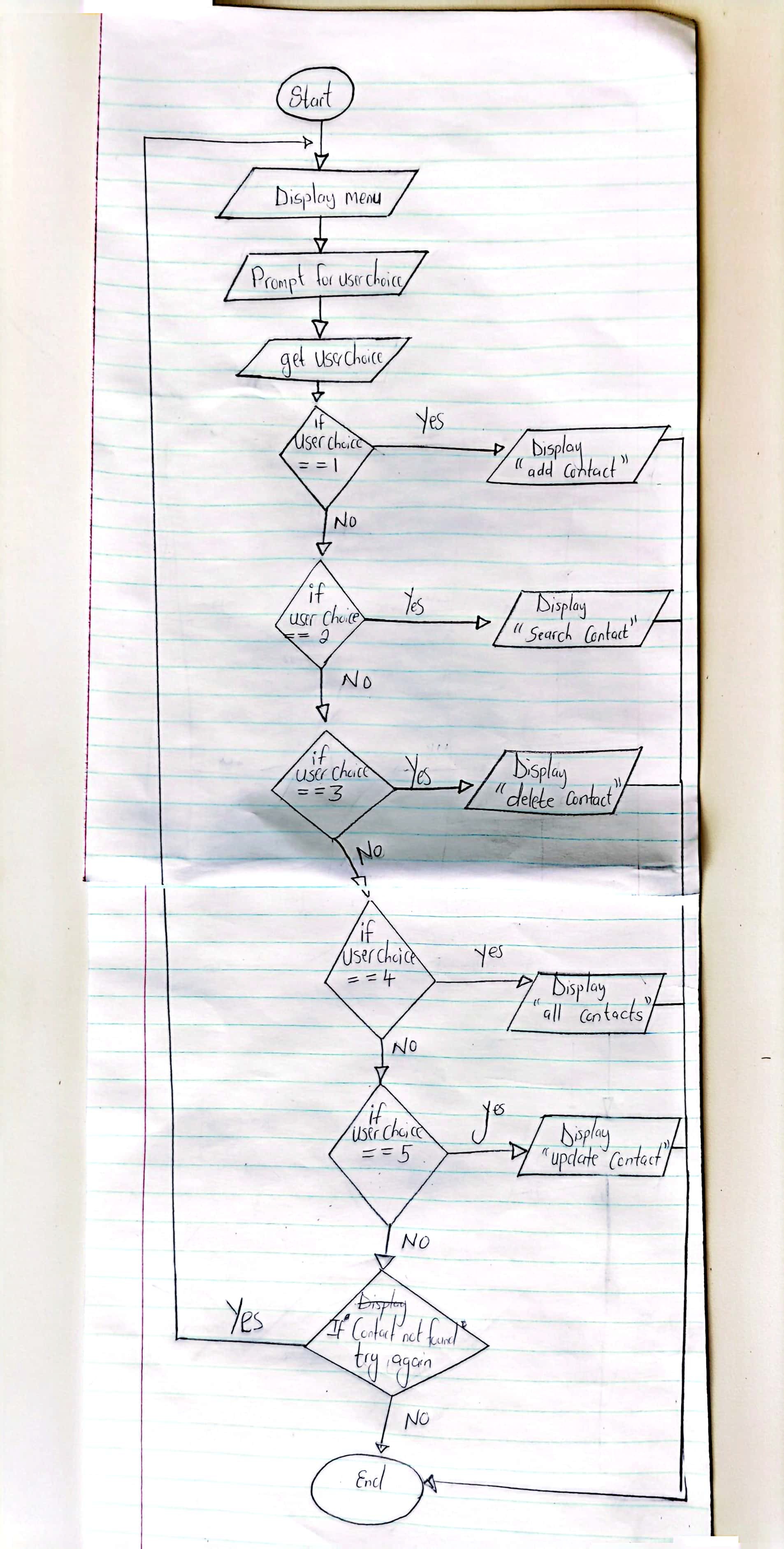
Break

Else:

Print "Invalid choice, please try again"

END

**The Flowchart**



**The Source Code**

