```
// Node Structure Module
class Node {
  name
  address
  phoneNumber
  next
  prev
}

// Phone Book Module
module PhoneBook {
  // Create The list
  head = Null

  // Function To Create Nodes For phone numbers
  function phoneBookListA() {
    Display "How many contacts are you adding?"
    Get numberOfContacts

    for (i = 1; i <= numberOfContacts; i++) {
      Prompt user for Name
      Get name
      Prompt user for Address
      Get address
      Prompt user for Phone Number
      Get phoneNumber

      Node newNode = new Node()
      newNode -> name = name
```

```
    newNode -> address = address
    newNode -> phoneNumber = phoneNumber

    if (head == Null) {
      newNode -> next = newNode
      newNode -> prev = newNode
      head = newNode
    } else {
      tail = head -> prev
      newNode -> next = head
      newNode -> prev = tail
      tail -> next = newNode
      head -> prev = newNode
      head = newNode
    }
  }
}

// Function to update a contact
function updateContact(nameToUpdate) {
  if head == Null {
    return "Contact list is empty"
  }
  temp = head
  do {
    if temp -> name == nameToUpdate {
      Prompt user for new Name
      Get newName
      temp -> name = newName
```

```
      Prompt user for new Address
      Get newAddress
      temp -> address = newAddress
      Prompt user for new Phone Number
      Get newPhoneNumber
      temp -> phoneNumber = newPhoneNumber
      return "Contact updated successfully"
    }
    temp = temp -> next
  } while (temp != head)
  return "Contact not found"
}

// Function to display all contacts
function displayContacts() {
  if head == Null {
    return "Contact list is empty"
  }
  temp = head
  do {
    Display "Name: " + temp -> name
    Display "Address: " + temp -> address
    Display "Phone Number: " + temp -> phoneNumber
    temp = temp -> next
  } while (temp != head)
}

// Function to delete a contact
function deleteFromPhoneBook() {
```

```
Display "How many contacts do you want to delete?"
Get contactsToBeDeleted

for (i = 1; i <= contactsToBeDeleted; i++) {
 if (head == Null) {
  return Display "List is Empty"
 }

 temp = head
 Display "Enter position to be deleted: "
 Get position

 // Validate position
 if (position < 1) {
  return Display "Invalid position"
 }

 for (j = 1; j < position; j++) {
  temp = temp -> next
  if (temp == head) { // Loop check
   return Display "Position out of bounds"
  }
 }

 prevNode = temp -> prev
 nextNode = temp -> next
 prevNode -> next = nextNode
 nextNode -> prev = prevNode
```

```
    if (temp == head) {
      head = nextNode
    }
    temp = free // Free the memory
  }
}

// Function to search for a contact by name
function searchContactByName(nameToFind) {
  if head == Null {
    return "Contact list is empty"
  }
  temp = head
  do {
    if temp -> name == nameToFind {
      return temp
    }
    temp = temp -> next
  } while (temp != head)

  return "Contact not found"
}

// Function to split the list
function splitList() {
  if (head == Null or head.next == head) {
    return head, Null
  }
```

```
  slow = head
  fast = head.next

  while (fast != head and fast.next != head) {
    slow = slow.next
    fast = fast.next.next
  }

  head1 = head
  head2 = slow.next
  slow.next = head1
  head2.prev = slow

  tail = head2.prev
  tail.next = head2

  return head1, head2
}

// Function to merge two lists
function mergeLists(head1, head2) {
  if (head1 == Null) {
    return head2
  }

  if (head2 == Null) {
    return head1
  }
```

```
    if (head1.name <= head2.name) {
      result = head1
      result.next = mergeLists(head1.next, head2)
      result.next.prev = result
    } else {
      result = head2
      result.next = mergeLists(head1, head2.next)
      result.next.prev = result
    }

    return result
  }

  // Function to sort the list using merge sort
  function mergeSort() {
    if (head == Null or head.next == head) {
      return head
    }

    head1, head2 = splitList()

    head1 = mergeSort(head1)
    head2 = mergeSort(head2)

    return mergeLists(head1, head2)
  }
}
```