Note: The Code file is attached separately.

Celebal Assignment Week – 2

Question: Create a Python program that implements a singly linked list using Object-Oriented Programming (OOP) principles. Your implementation should include the following: A Node class to represent each node in the list. A Linked List class to manage the nodes, with methods to: Add a node to the end of the list Print the list Delete the nth node (where n is a 1-based index) Include exception handling to manage edge cases such as: Deleting a node from an empty list Deleting a node with an index out of range Test your implementation with at least one sample list.

Solution-

Program has been created to handle add, print and delete through menu. Below errors and exceptions are also taken care of.

Errors and Exceptions Handled:

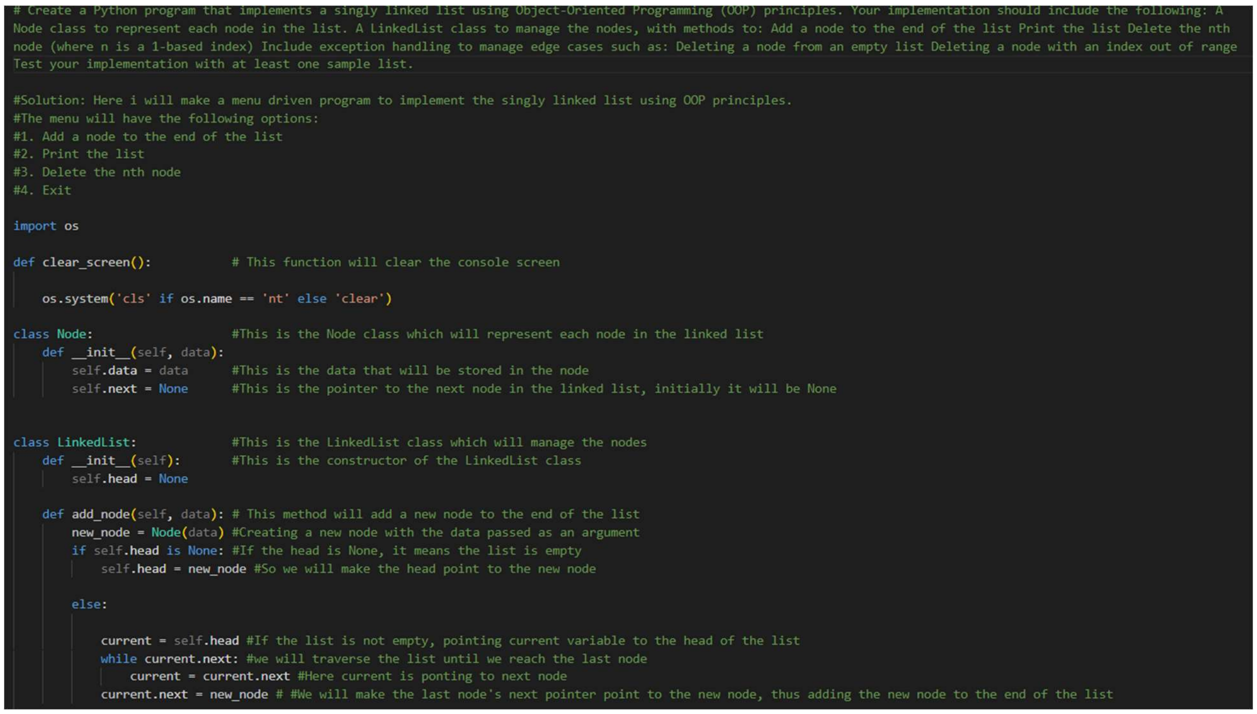
- Sample list with 4 nodes and then provision to add more nodes at the end of the linked list

- Empty List Deletion check
  - Out of range index check
  - Negative index check
  - Empty list check
  - Menu control with proper messages

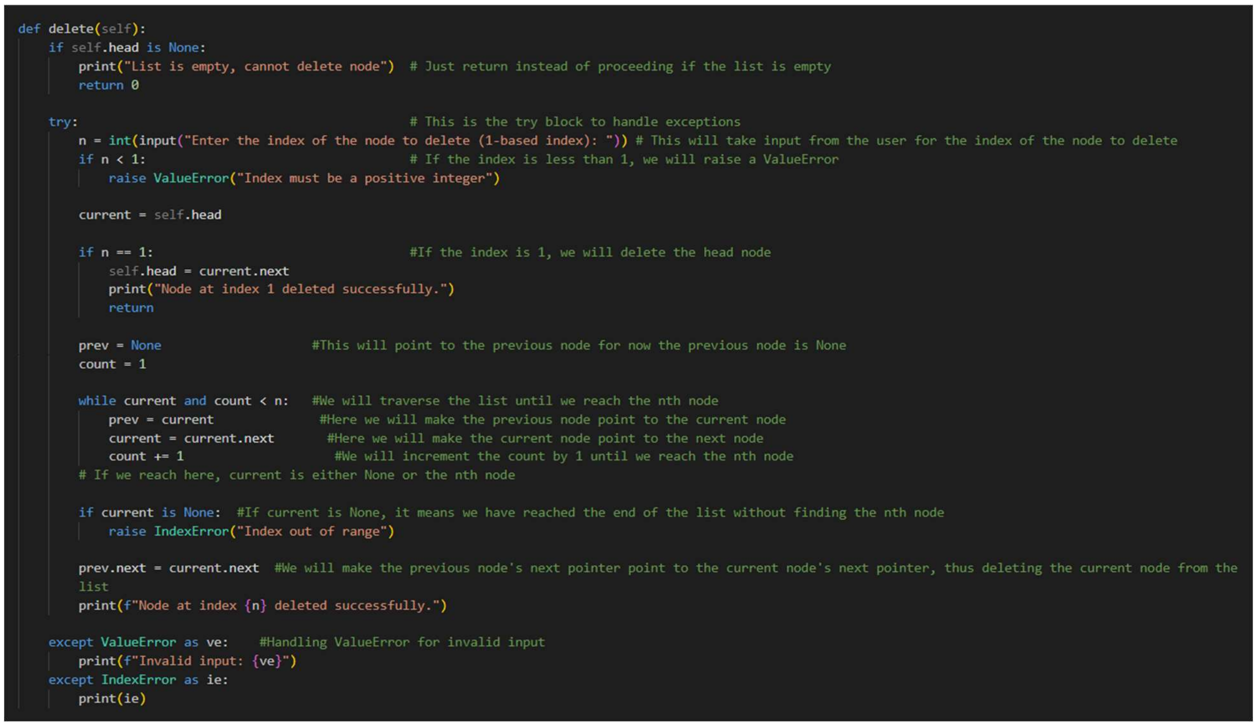
Algorithm:

- Define Node Class with data and a next pointer to link nodes
- **Define** Linked List **class** with a head to track the start of the list.
- **Add node method**: Traverse to the end and link a new node there.
- **Print method**: Traverse from head to end, printing each node's data.
- **Delete nth node method**: Validate index, then unlink the nth node by adjusting pointers.
- **Handle edge cases**: Raise/handle exceptions for empty list or out-of-range index.
- **Pre-load list** with sample data: 10 → 20 → 30 → 40 (user can add more)
- **Display menu** with options to add, print, delete, or exit.
- **Add node flow:** Prompt repeatedly to add new nodes until user declines
- **Delete node flow:** Prompt index and delete that node, validating input and state

Screenshots of the Solution:

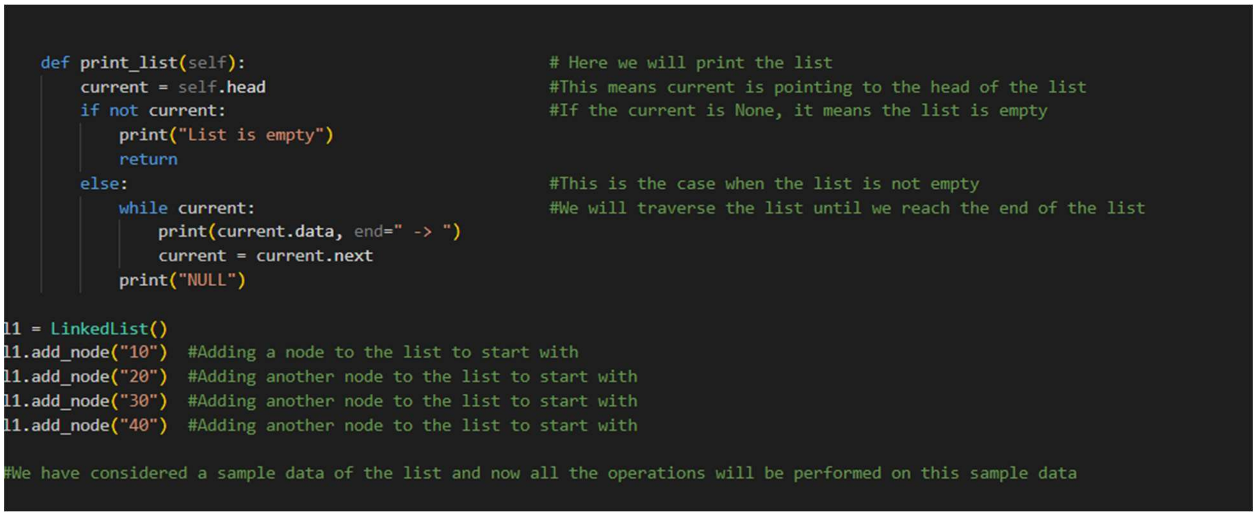


```python
# Create a Python program that implements a singly linked list using Object-Oriented Programming (OOP) principles. Your implementation should include the following: A
Node class to represent each node in the list. A LinkedList class to manage the nodes, with methods to: Add a node to the end of the list Print the list Delete the nth
node (where n is a 1-based index) Include exception handling to manage edge cases such as: Deleting a node from an empty list Deleting a node with an index out of range
Test your implementation with at least one sample list.

#Solution: Here i will make a menu driven program to implement the singly linked list using OOP principles.
#The menu will have the following options:
#1. Add a node to the end of the list
#2. Print the list
#3. Delete the nth node
#4. Exit

import os

def clear_screen():           # This function will clear the console screen

    os.system('cls' if os.name == 'nt' else 'clear')

class Node:                   #This is the Node class which will represent each node in the linked list
    def __init__(self, data):
        self.data = data      #This is the data that will be stored in the node
        self.next = None      #This is the pointer to the next node in the linked list, initially it will be None


class LinkedList:             #This is the LinkedList class which will manage the nodes
    def __init__(self):       #This is the constructor of the LinkedList class
        self.head = None

    def add_node(self, data): # This method will add a new node to the end of the list
        new_node = Node(data) #Creating a new node with the data passed as an argument
        if self.head is None: #If the head is None, it means the list is empty
            self.head = new_node #So we will make the head point to the new node

        else:

            current = self.head #If the list is not empty, pointing current variable to the head of the list
            while current.next: #we will traverse the list until we reach the last node
                current = current.next #Here current is ponting to next node
            current.next = new_node # #We will make the last node's next pointer point to the new node, thus adding the new node to the end of the list
```

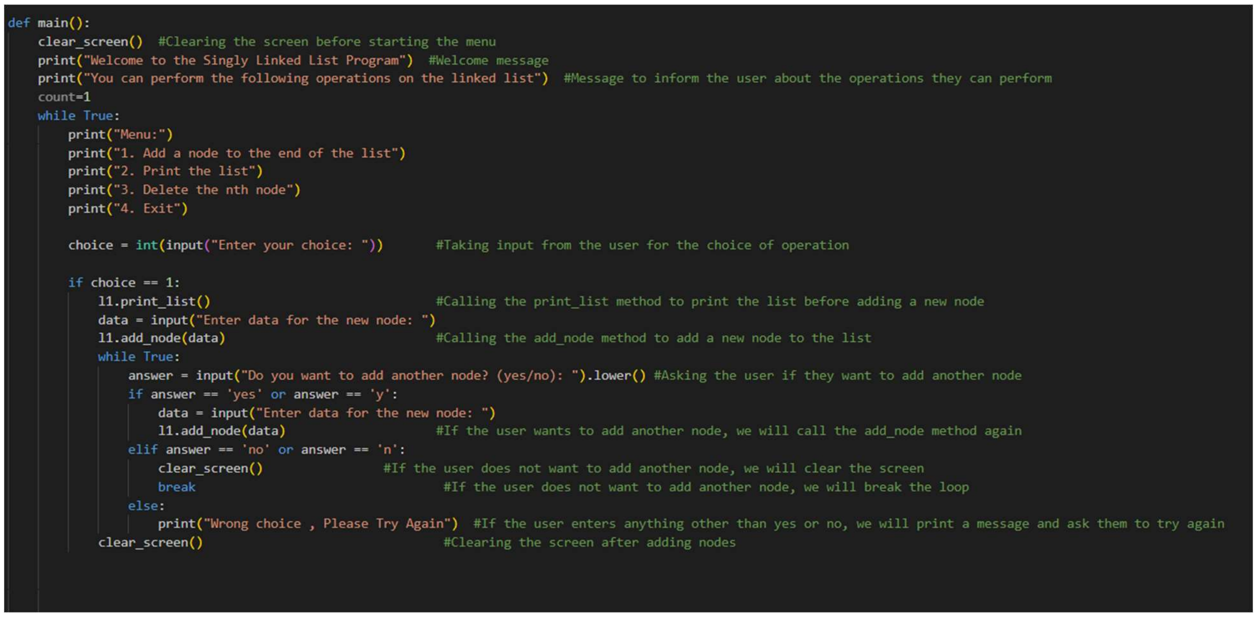


```python
def delete(self):
    if self.head is None:
        print("List is empty, cannot delete node")  # Just return instead of proceeding if the list is empty
        return 0

    try:                                            # This is the try block to handle exceptions
        n = int(input("Enter the index of the node to delete (1-based index): ")) # This will take input from the user for the index of the node to delete
        if n < 1:                                   # If the index is less than 1, we will raise a ValueError
            raise ValueError("Index must be a positive integer")

        current = self.head

        if n == 1:                                  #If the index is 1, we will delete the head node
            self.head = current.next
            print("Node at index 1 deleted successfully.")
            return

        prev = None                     #This will point to the previous node for now the previous node is None
        count = 1

        while current and count < n:    #We will traverse the list until we reach the nth node
            prev = current               #Here we will make the previous node point to the current node
            current = current.next        #Here we will make the current node point to the next node
            count += 1                      #We will increment the count by 1 until we reach the nth node
        # If we reach here, current is either None or the nth node

        if current is None:  #If current is None, it means we have reached the end of the list without finding the nth node
            raise IndexError("Index out of range")

        prev.next = current.next  #We will make the previous node's next pointer point to the current node's next pointer, thus deleting the current node from the
        list
        print(f"Node at index {n} deleted successfully.")

    except ValueError as ve:    #Handling ValueError for invalid input
        print(f"Invalid input: {ve}")
    except IndexError as ie:
        print(ie)
```

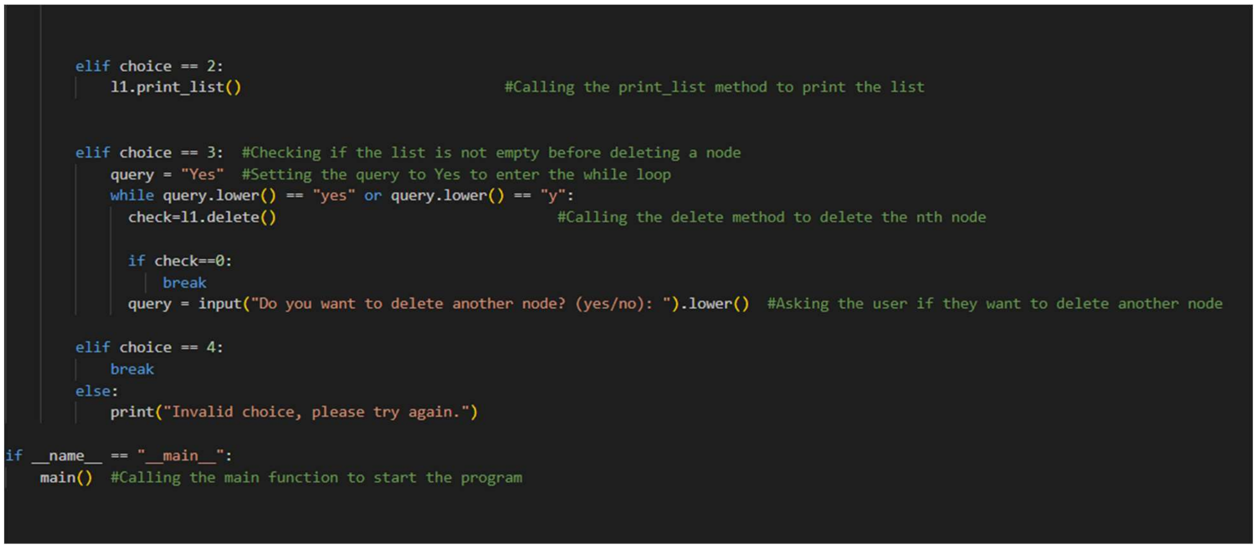
Tushar Nag



```
    def print_list(self):                              # Here we will print the list
        current = self.head                            #This means current is pointing to the head of the list
        if not current:                                #If the current is None, it means the list is empty
            print("List is empty")
            return
        else:                                          #This is the case when the list is not empty
            while current:                             #We will traverse the list until we reach the end of the list
                print(current.data, end=" -> ")
                current = current.next
            print("NULL")

l1 = LinkedList()
l1.add_node("10")  #Adding a node to the list to start with
l1.add_node("20")  #Adding another node to the list to start with
l1.add_node("30")  #Adding another node to the list to start with
l1.add_node("40")  #Adding another node to the list to start with

#We have considered a sample data of the list and now all the operations will be performed on this sample data
```



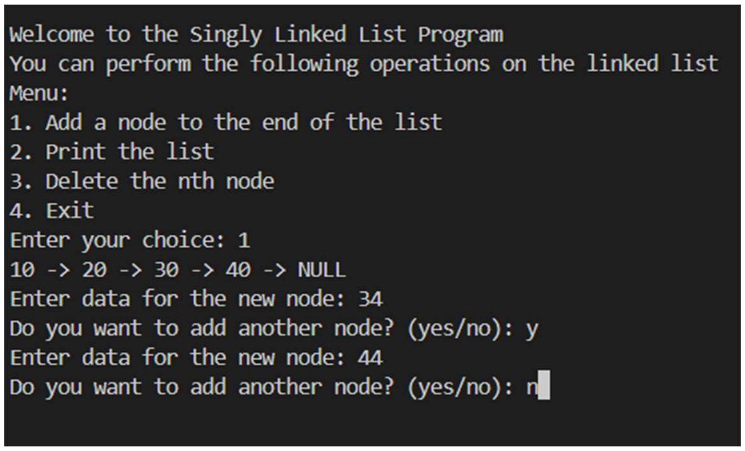
```
def main():
    clear_screen()  #Clearing the screen before starting the menu
    print("Welcome to the Singly Linked List Program")  #Welcome message
    print("You can perform the following operations on the linked list")  #Message to inform the user about the operations they can perform
    count=1
    while True:
        print("Menu:")
        print("1. Add a node to the end of the list")
        print("2. Print the list")
        print("3. Delete the nth node")
        print("4. Exit")

        choice = int(input("Enter your choice: "))       #Taking input from the user for the choice of operation

        if choice == 1:
            l1.print_list()                              #Calling the print_list method to print the list before adding a new node
            data = input("Enter data for the new node: ")
            l1.add_node(data)                            #Calling the add_node method to add a new node to the list
            while True:
                answer = input("Do you want to add another node? (yes/no): ").lower() #Asking the user if they want to add another node
                if answer == 'yes' or answer == 'y':
                    data = input("Enter data for the new node: ")
                    l1.add_node(data)                    #If the user wants to add another node, we will call the add_node method again
                elif answer == 'no' or answer == 'n':
                    clear_screen()                #If the user does not want to add another node, we will clear the screen
                    break                                 #If the user does not want to add another node, we will break the loop
                else:
                    print("Wrong choice , Please Try Again")  #If the user enters anything other than yes or no, we will print a message and ask them to try again
            clear_screen()                                #Clearing the screen after adding nodes
```



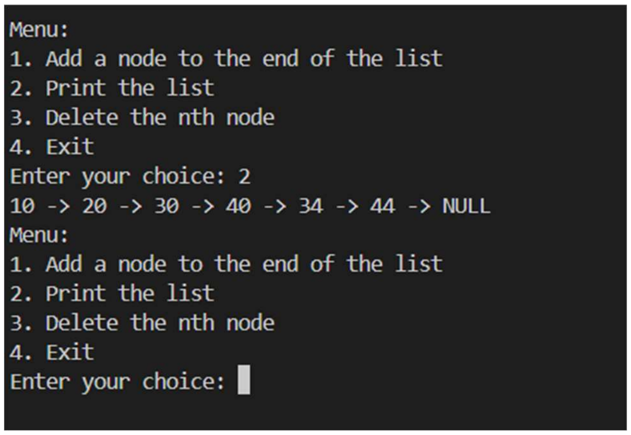
```
        elif choice == 2:
            l1.print_list()                              #Calling the print_list method to print the list


        elif choice == 3:  #Checking if the list is not empty before deleting a node
            query = "Yes"  #Setting the query to Yes to enter the while loop
            while query.lower() == "yes" or query.lower() == "y":
                check=l1.delete()                        #Calling the delete method to delete the nth node

                if check==0:
                    break
                query = input("Do you want to delete another node? (yes/no): ").lower()  #Asking the user if they want to delete another node

        elif choice == 4:
            break
        else:
            print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()  #Calling the main function to start the program
```

Tushar Nag

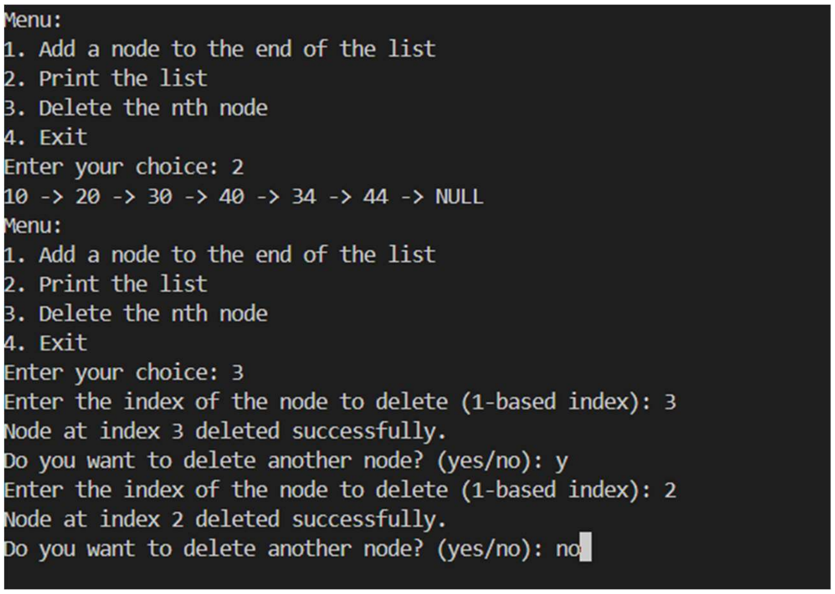
Output:



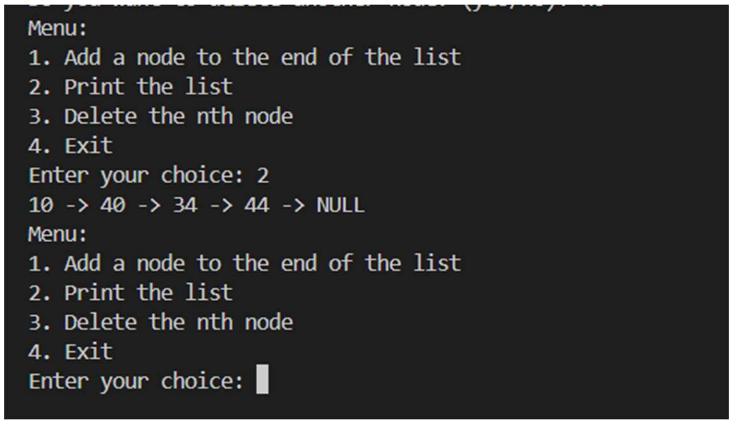
```
Welcome to the Singly Linked List Program
You can perform the following operations on the linked list
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice: 1
10 -> 20 -> 30 -> 40 -> NULL
Enter data for the new node: 34
Do you want to add another node? (yes/no): y
Enter data for the new node: 44
Do you want to add another node? (yes/no): n
```



```
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice: 2
10 -> 20 -> 30 -> 40 -> 34 -> 44 -> NULL
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice:
```



```
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice: 2
10 -> 20 -> 30 -> 40 -> 34 -> 44 -> NULL
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice: 3
Enter the index of the node to delete (1-based index): 3
Node at index 3 deleted successfully.
Do you want to delete another node? (yes/no): y
Enter the index of the node to delete (1-based index): 2
Node at index 2 deleted successfully.
Do you want to delete another node? (yes/no): no
```



```
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice: 2
10 -> 40 -> 34 -> 44 -> NULL
Menu:
1. Add a node to the end of the list
2. Print the list
3. Delete the nth node
4. Exit
Enter your choice: 
```

Tushar Nag