Lets upgrade

Assignment 4

Name:- Apurv Aglawe

Email:- itsmyidapurv@gmail.com

Question 1:

Implement deletion operation from the end of the linked list and Insertion operation from the beginning of the linked list.

Solution:-

```
class Node:
    def __init__(self,data):
        self.data = data
        self.ref = None
class LinkedList:
    def __init__(self):
        self.head = None

    def print_List(self):
        if self.head is None:
            print("linked list in empty")
        else:
            n = self.head
            while n is not None:
                print(n.data)
                n = n.ref
    # Function to add elements from the beginning
    def add_beginning(self,data):
        new_node = Node(data)
        new_node.ref = self.head
        self.head = new_node

    # Function to delete element from the end
    def del_end(self):
        if self.head is None:
            print("linked list in empty")
        else:
            n = self.head
            while n.ref.ref is not None:
                n = n.ref
            n.ref = None
```

```
LL1 = LinkedList()
# Add elements 10, 20, 30, 40 from the beginning
LL1.add_beginning(10)
LL1.add_beginning(20)
LL1.add_beginning(30)
LL1.add_beginning(40)
# Delete the last element
LL1.del_end()
# print the list after addition and deletion operations
LL1.print_List()
```

Question 2

Implement binary search using python language.
(Write a function which returns the index of x in given array arr if present, else returns -1)

Solution:-

```
arr = [ int(i) for i in input("Enter the values of the array: ").split() ]
n = int( input("Find the element to be searched: "))
def find_index(n):
    for i in range(0,len(arr)):
        if arr[i] == n:
            return i
    else:
        return -1
find_index(n)
```

Question 3

Write a Python program to find the middle of a linked list.

Solution:-

```
class Node:
    def __init__(self,data):
        self.data = data
        self.ref = None
class LinkedList:
    def __init__(self):
        self.head = None

    def print_List(self):
        if self.head is None:
            print("linked list in empty")
        else:
            n = self.head
            while n is not None:
                print(n.data)
```

```
                n = n.ref
# Function to print mid value of the linked list
    def print_Mid(self):
        temp = self.head
        count = 0
        while self.head is not None:
            if(count %2 != 0):
                temp = temp.ref
            self.head = self.head.ref
            count += 1
        print("The mid element is:", temp.data)

LL1 = LinkedList()
# Add elements 10, 20, 30, 40, 50, 60, 70 from the beginning
LL1.add_beginning(10)
LL1.add_beginning(20)
LL1.add_beginning(30)
LL1.add_beginning(40)
LL1.add_beginning(50)
LL1.add_beginning(60)
LL1.add_beginning(70)
# Print the Middle element of the linked list
LL1.print_Mid()
```