LetsUpgrade Data Structure and Algorithm Essentials
Assignment 4 | 21st January 2021

Question 1

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

Answer:

```
class Node:
    def __init__(self, data):
        self.item = data
        self.ref = None
class LinkedList:
    def __init__(self):
        self.start_node = None
    def traverse_list(self):
        if self.start_node is None:
            print("List has no element")
            return
        else:
            n = self.start_node
            while n is not None:
                print(n.item , " ")
                n = n.ref
    def insert_at_start(self, data):
        new_node = Node(data)
        new_node.ref = self.start_node
        self.start_node= new_node
    def delete_at_end(self):
        if self.start_node is None:
            print("The list has no element to delete")
            return
        n = self.start_node
```

```
        while n.ref.ref is not None:
            n = n.ref
        n.ref = None
new_linked_list = LinkedList()
print("Insertion operation from the beginning of the linked list:")
new_linked_list.insert_at_start(20)
new_linked_list.insert_at_start(30)
new_linked_list.insert_at_start(40)
new_linked_list.traverse_list()
print("deletion operation from the end of the linked list:")
new_linked_list.delete_at_end()
new_linked_list.traverse_list()
```

Output:

```
Insertion operation from the beginning of the linked list:
40
30
20
deletion operation from the end of the linked list:
40
30
```

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)

Answer:

```
def binary_search(arr, low, high, x):
    if high >= low:
```

```
        mid = (high + low) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)
        else:
            return binary_search(arr, mid + 1, high, x)
    else:
        return -1
arr = [ 2, 3, 4, 10, 40 ]
x = 10
result = binary_search(arr, 0, len(arr)-1, x)
if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")
```

Output:

Element is present at index 3

Question 3

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

Answer:

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

```
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
    def printMiddle(self):
        slow_ptr = self.head
        fast_ptr = self.head
        if self.head is not None:
            while (fast_ptr is not None and fast_ptr.next is not None):
                fast_ptr = fast_ptr.next.next
                slow_ptr = slow_ptr.next
            print("The middle element is: ", slow_ptr.data)
list1 = LinkedList()
list1.push(5)
list1.push(4)
list1.push(2)
list1.push(3)
list1.push(1)
list1.printMiddle()
```

Output:

```
The middle element is:  2
```

Submitted To: Subrat Kumar Swain *(LetsUpgrade Instructor)*
Submitted By: Arun Sharma
curiousarun08@gmail.com